Large Language Models

Generative Pre-trained Transformer

GPT

Developed by OpenAI (participated by Microsoft)

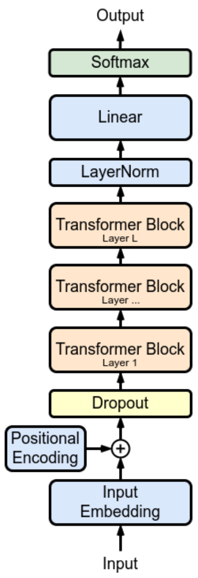
ChatGPT is the last of this family

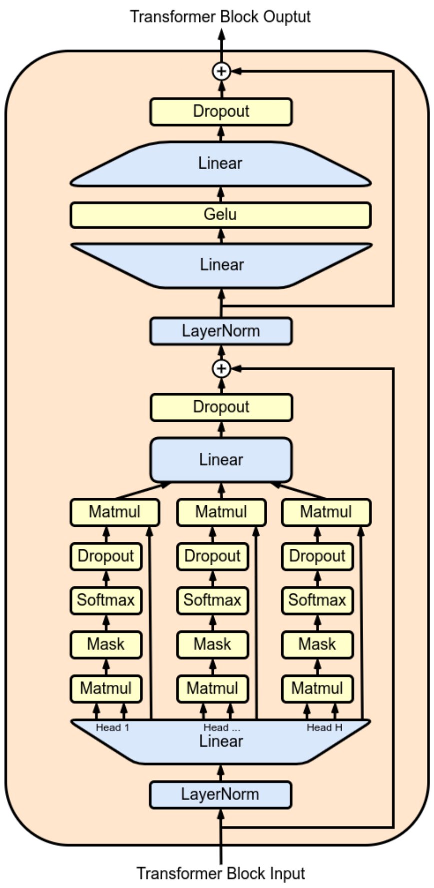
- Pre-trained with Language Modeling then fine-tuned with supervision

GPT

The original GPT model is a stack of 12 *transformer decoder* blocks

- Variant of the transformer without the encoder part
- As in BERT, the FFN at the end of each block uses GeLU activation





Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.

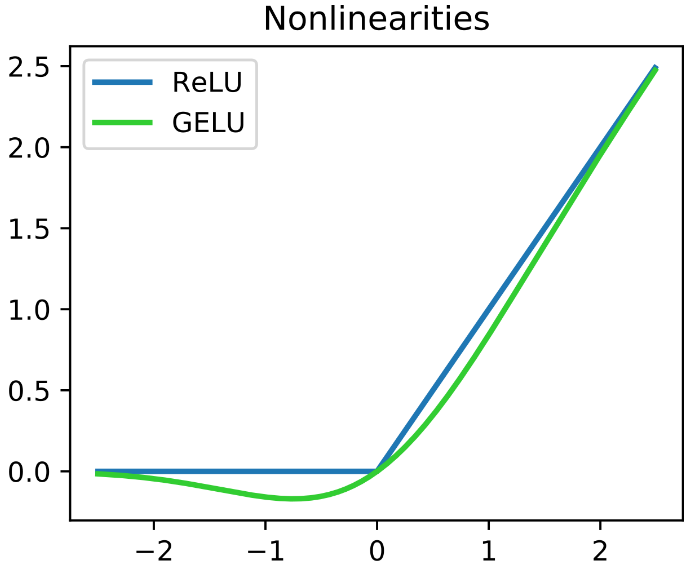
GeLU activation function

Gaussian Error Linear Unit is a variant of ReLU where the standard Gaussian cumulative distribution is used to modulate the linear activation

$$\mathrm{GELU}(x) = xP(X \leq x)$$

$$X \sim \mathcal{N}(0, 1)$$

$$\mathrm{GELU}(x) \sim x\sigma(1.702x)$$



https://bit.ly/3sNBObx

GPT unsupervised pre-training

Given $\mathcal{U} = \{u_1, \ldots, u_n\}$ a set of tokens:

Token embedding

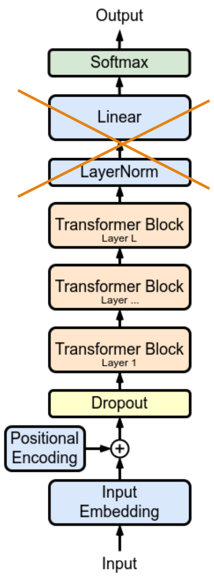
position embedding

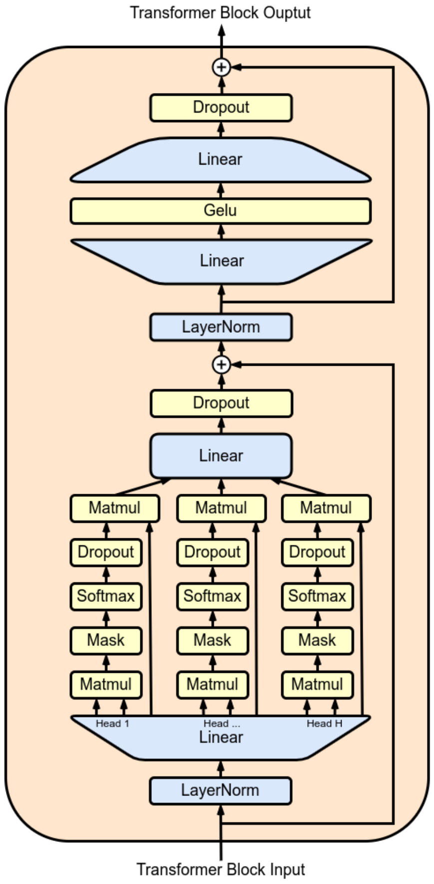
$$h_0 = U\boxed{W_e} + \boxed{W_p}$$

$$h_l = \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n]$$

$$P(u) = \texttt{softmax}(h_n W_e^T)$$

where $U = (u_{-k}, \ldots, u_{-1})$ is the context





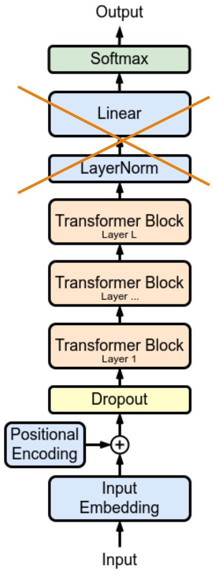
Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.

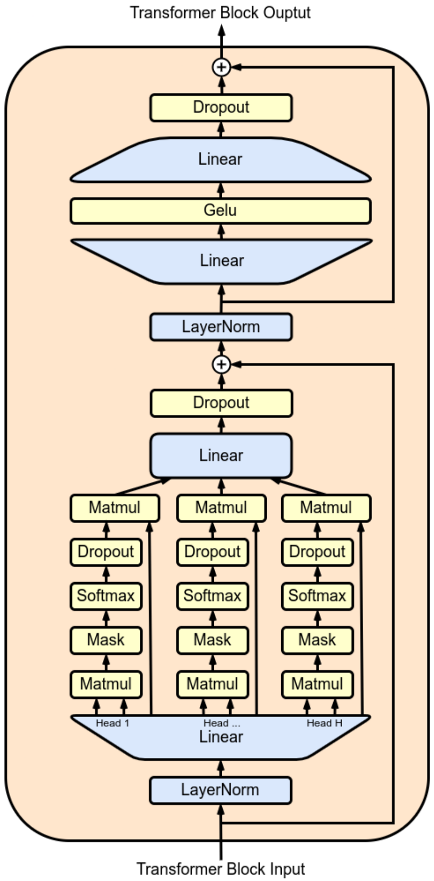
GPT unsupervised pre-training

$$h_0 = UW_e + W_p$$
$$h_l = \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n]$$
$$P(u) = \texttt{softmax}(h_n W_e^T)$$

Language Modeling objective function:

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$





Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.

GPT supervised fine-tuning

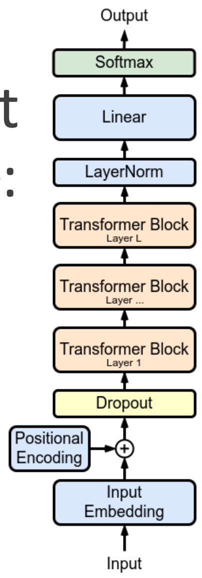
A linear layer is added on top to predict a label *y* from a set of tokens $\{x_1, \ldots, x_n\}$:

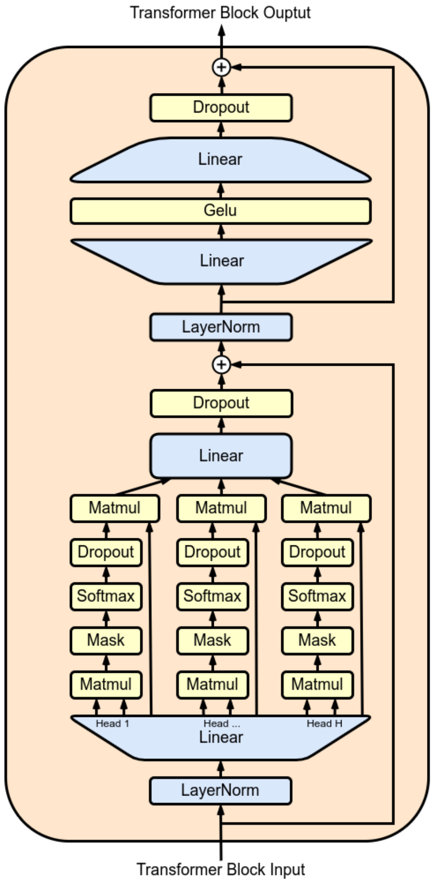
$$P(y|x^1, \ldots, x^m) = \texttt{softmax}(h_l^m W_y).$$

with a composite objective function:

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C})$$

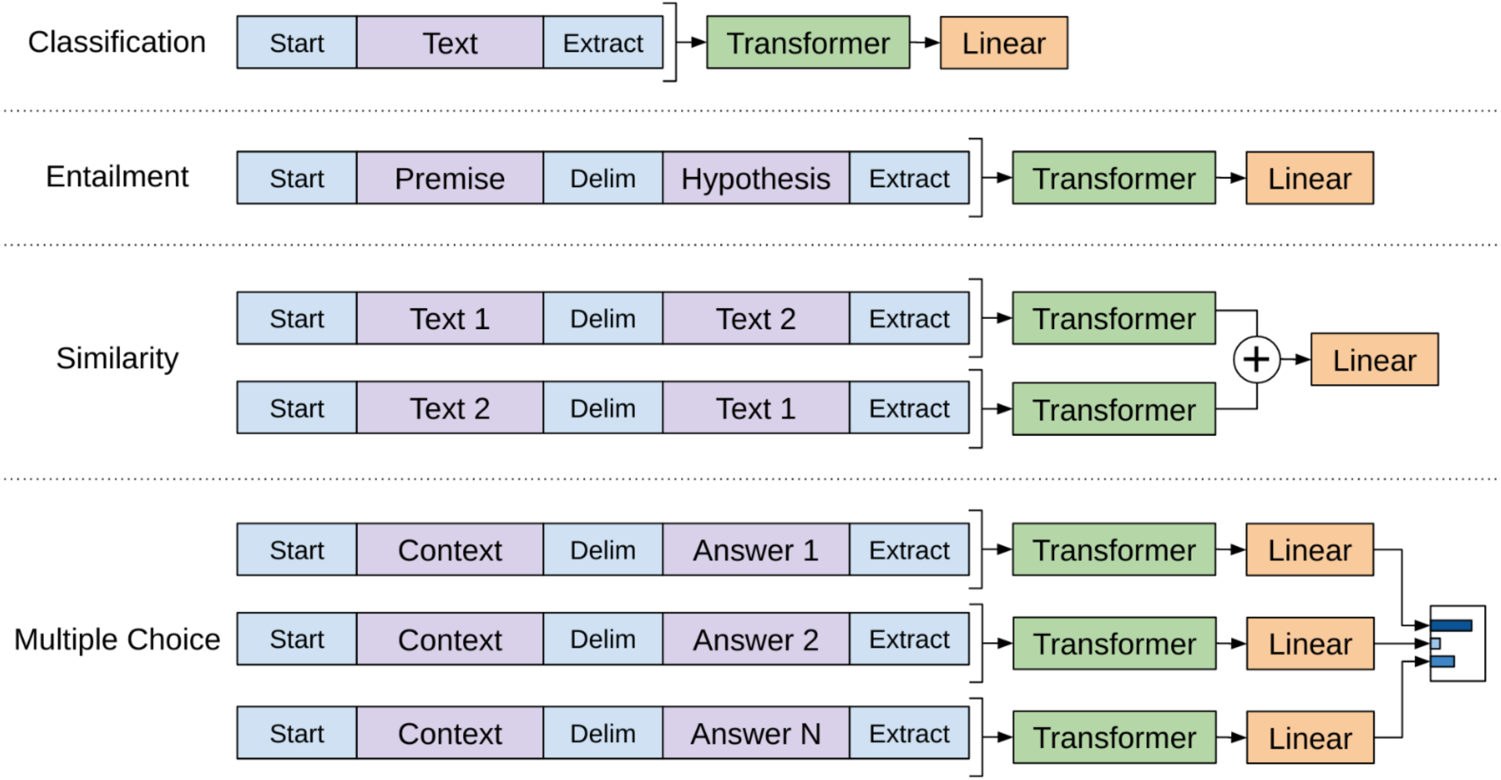
$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \ldots, x^m)$$





Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.

GPT fine-tuning tasks



Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018).
Improving language understanding by generative pre-training.

GPT families

Model	Architecture	Parameter count	Training data	Release date	Training cost
GPT-1	12-level, 12-headed Transformer decoder (no encoder), followed by linear-softmax.	117 million	BookCorpus:[27] 4.5 GB of text, from 7000 unpublished books of various genres.	June 11, 2018[8]	30 days on 8 P600 GPUs, or 1 petaFLOP/s-day.[8]
GPT-2	GPT-1, but with modified normalization	1.5 billion	WebText: 40 GB of text, 8 million documents, from 45 million webpages upvoted on Reddit.	February 14, 2019 (initial/limited version) and November 5, 2019 (full version)[28]	"tens of petaflop/s-day",[29] or 1.5e21 FLOP.[30]
GPT-3	GPT-2, but with modification to allow larger scaling	175 billion[31]	499 billion tokens consisting of CommonCrawl (570 GB), WebText, English Wikipedia, and two books corpora (Books1 and Books2).	May 28, 2020[29]	3640 petaflop/s-day (Table D.1 [29]), or 3.1e23 FLOP.[30]
GPT-3.5	Undisclosed	175 billion[31]	Undisclosed	March 15, 2022	Undisclosed
GPT-4	Also trained with both text prediction and RLHF; accepts both text and images as input. Further details are not public.[26]	Undisclosed. Estimated 1.7 trillion[32]	Undisclosed	March 14, 2023	Undisclosed. Estimated 2.1e25 FLOP.[30]

Wikipedia

ChatGPT

Based on GPT-3.5 and GPT-4

- Fine-tuned to target conversational usage
- Uses the so called *Reinforcement Learning With Human Feedback* (RLHF)

RLHF

Makes use of human trainers to improve model performance

- Human trainers rank the response provided by the model in a previous conversation

- Ranks are then used to create a reward model used in the iterations of the *Proximal Policy Optimization* (PPO) reinforcement learning algorithm

LLaMA

Large Language Model Meta AI was released by Meta in February 2023

- 7, 16, 33, and 65B parameters versions
- 13B parameters was reported to outperform GPT-3
- LLaMA-2 released in July 2023 with 7, 13, and 70B versions

https://bit.ly/3Rf4K5C

LLaMA-2

Based on transformer decoder stack

Minor architectural differences with GPT-3:

- SwiGLU activation function instead of ReLU
- Rotary positional embeddings
- Root-mean-squared layer-normalization instead of standard layer normalization
- Increases context length to 4K tokens

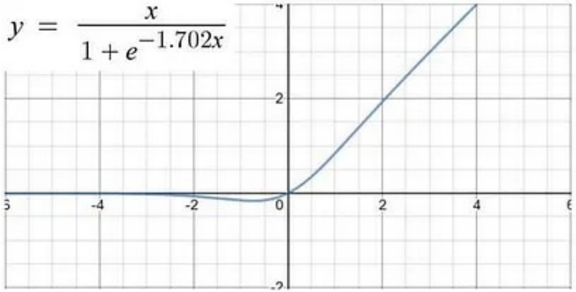
https://bit.ly/3Rf4K5C

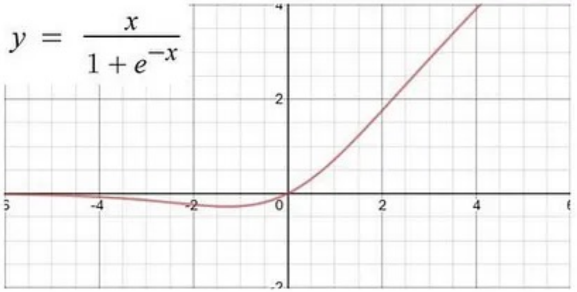
SwiGLU activation function

Swish Gated Linear Unit has an activation function that is the combination of Swish and Gated LUs

Swish LU are a generalization of the GELU approximation

$$\text{Swish}(x) = x \cdot \sigma(\beta x)$$





https://bit.ly/40T9npg

SwiGLU activation function

Gated LUs embed a linear activation inside the sigmoid function

$$\mathrm{GLU}(x) = \sigma(Wx + b)$$

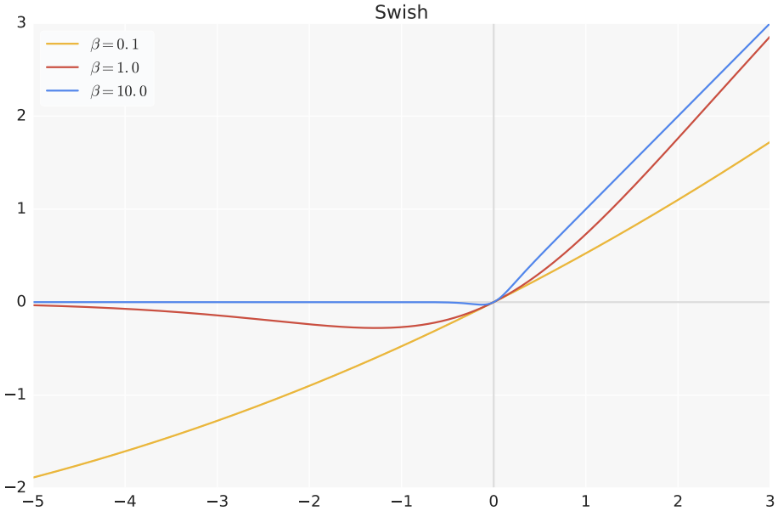
Gating mechanism: the neuron is activated based on the input it receives

https://bit.ly/40T9npg

SwiGLU activation function

SwiGLU embeds the previous activations

$$\mathrm{SwiGLU}(x) = x \cdot \sigma(\beta x) + (1 - \sigma(\beta x)) \cdot \sigma(W \cdot x + b)$$



https://bit.ly/40T9npg

Rotary Positional Embeddings (RoPE)

A particular type of *relative position embeddings*

- We can represent **q**, **k**, and **v** self-attention vectors in terms of their embeddings as

$$\begin{aligned} \boldsymbol{q}_m &= f_q(\boldsymbol{x}_m, m) \\ \boldsymbol{k}_n &= f_k(\boldsymbol{x}_n, n) \\ \boldsymbol{v}_n &= f_v(\boldsymbol{x}_n, n), \end{aligned}$$

- The element (m, n) of the attention matrix $\mathbf{q}_m \mathbf{k}_n$ is formulated in terms of relative position $m - m$ as a function $g(\boldsymbol{x}_m, \boldsymbol{x}_n, m - n)$

Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., & Liu, Y. (2021). Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*.

Rotary Positional Embeddings (RoPE)

A particular type of *relative position embeddings*

- RoPE expresses *g* as

$$f_q(\boldsymbol{x}_m, m) = (\boldsymbol{W}_q \boldsymbol{x}_m) e^{im\theta}$$
$$f_k(\boldsymbol{x}_n, n) = (\boldsymbol{W}_k \boldsymbol{x}_n) e^{in\theta}$$
$$g(\boldsymbol{x}_m, \boldsymbol{x}_n, m-n) = \mathrm{Re}[(\boldsymbol{W}_q \boldsymbol{x}_m)(\boldsymbol{W}_k \boldsymbol{x}_n)^* e^{i(m-n)\theta}]$$

- So the **q** and **k** vectors, and their inner product are

$$f_{\{q,k\}}(\boldsymbol{x}_m, m) = \boldsymbol{R}^d_{\Theta,m} \boldsymbol{W}_{\{q,k\}} \boldsymbol{x}_m$$
$$\boldsymbol{q}_m^\intercal \boldsymbol{k}_n = (\boldsymbol{R}^d_{\Theta,m} \boldsymbol{W}_q \boldsymbol{x}_m)^\intercal (\boldsymbol{R}^d_{\Theta,n} \boldsymbol{W}_k \boldsymbol{x}_n) = \boldsymbol{x}_m^\intercal \boldsymbol{W}_q R^d_{\Theta,n-m} \boldsymbol{W}_k \boldsymbol{x}_n$$
$$\Theta = \{\theta_i = 10000^{-2(i-1)/d}, i \in [1, 2, ..., d/2]\}$$

Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., & Liu, Y. (2021). Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*.

LLaMA training

LLaMA-1 trained with 1.4 trillion tokens

- Webpages scraped by CommonCrawl
- Open source repositories of source code from GitHub
- Wikipedia in 20 different languages
- Public domain books from Project Gutenberg
- The LaTeX source code for scientific papers uploaded to ArXiv
- Questions and answers from Stack Exchange websites

https://bit.ly/3Rf4K5C

LLaMA training

LLaMA-2 trained with 2 trillion tokens

LLaMA-2 chat was fine-tuned on 27,540 prompt-response pairs created for this project

RLHF was used with rejection sampling and PPO

https://bit.ly/3Rf4K5C

Large Language Models

Conditioning LLMs for downstream applications

Conditioning techniques

GPT based LLMs do not own knowledge about too specific topics

- They have been trained
  - with documents until the 2021/2022
  - on corpora dealing with «general» topics
- Hallucinations

Conditioning techniques

There are two main contitioning techniques to contrast hallucination, and achieving precise answers

- Fine-tuning
- Retrieval Augmented Generation (RAG)

Fine-tuning

LLMs require too much GPU memory space

- Approximately a number of GB that is 2 times the number of parameters
  - Normally 16bit floats are used for inference
- An example: LLaMA-2 7B → up to 16GB including embeddings

https://bit.ly/3uxVpwX

Fine-tuning

LLMs require too much GPU memory space

- When fine-tuning LLaMA-2 7B we have to take into account the following (with batch size = 1)
- Model parameters (16bit floats): 7B x 2 = 14GB
- Gradients (16bit floats): 7B x 2 = 14GB
- Optimizer states (2 x parameter 32bit floats): 7B x 4 x 2 = 56GB

TOTAL: 84GB

https://bit.ly/47tcy9v

Low Rank Adaptation (LoRA)

Agnostic with respect to both the model and the training objective

Assume to have

- a set of trained parameters Φ_0 that have to be fine-tuned to $\Phi_0 + \Delta\Phi$
- A context-target pairs dataset $\mathcal{Z} = \{(x_i, y_i)\}_{i = 1, \ldots, N}$
- We have to maximize: $$\max_{\Phi} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log\left(P_{\Phi}(y_t | x, y_{<t})\right)$$

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... & Chen, W. (2021). Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Low Rank Adaptation (LoRA)

LoRa tries to maximize $\max_{\Theta} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log\left(p_{\Phi_0+\Delta\Phi(\Theta)}(y_t|x, y_{<t})\right)$

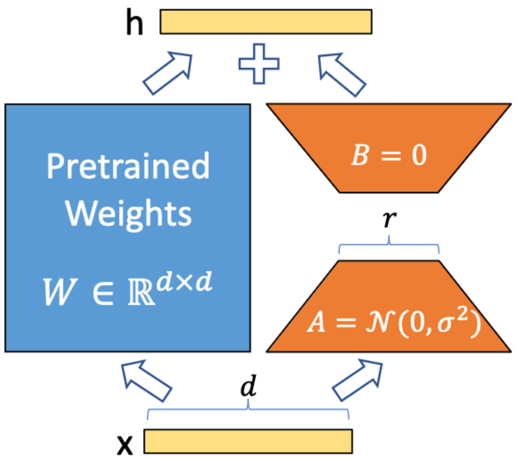
$$\Delta\Phi = \Delta\Phi(\Theta) \quad |\Theta| \ll |\Phi_0|$$

For a generic hidden layer:

$$h = W_0 x + \Delta W x = W_0 x + BAx$$

$W_0 \in \mathbb{R}^{d\times k}$, $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$

$r << \min(d, k)$



Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... & Chen, W. (2021). Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Retrieval Augmented Generation (RAG)

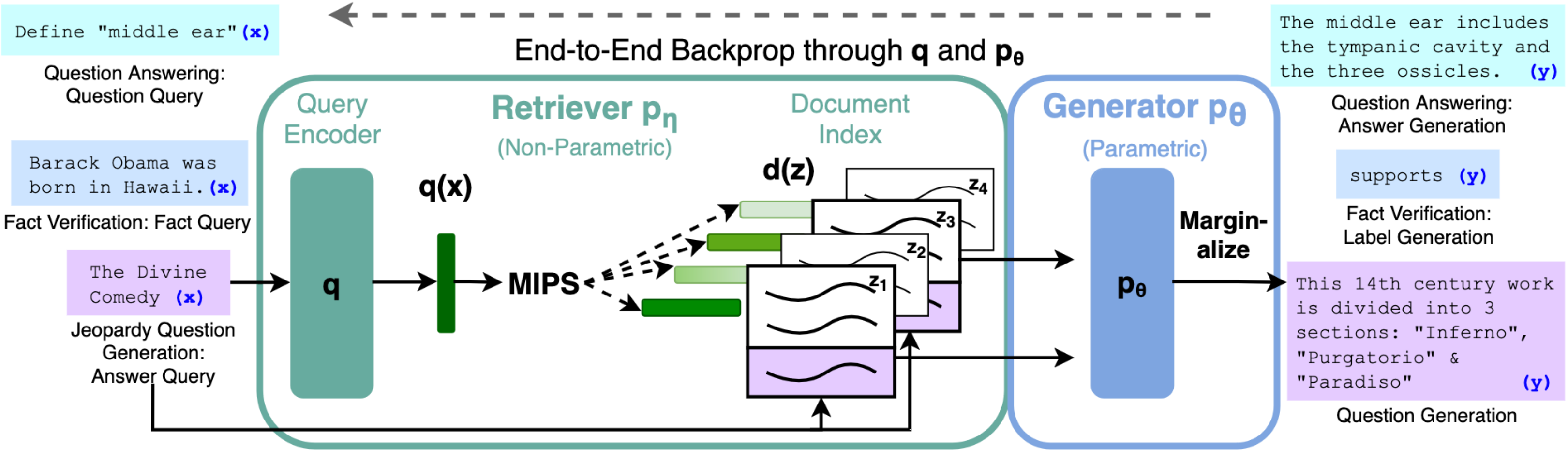
A technique to couple task-specific document retrieval with language generation to achieve precise answers

- Used mainly in QA with the chat versions of GPT LLMs
- Can be applied to whatever downstream application
- No (or at least very little) fine-tuning

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, *33*, 9459-9474.

Retrieval Augmented Generation (RAG)

A *parametric memory* (sequence-to-sequence LM) is complemented with a *non parametric memory* (a document retriever using vector embeddings)



Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, *33*, 9459-9474.

Retrieval Augmented Generation (RAG)

$$p_\eta(z|x) \propto \exp\left(\mathbf{d}(z)^\top \mathbf{q}(x)\right)$$

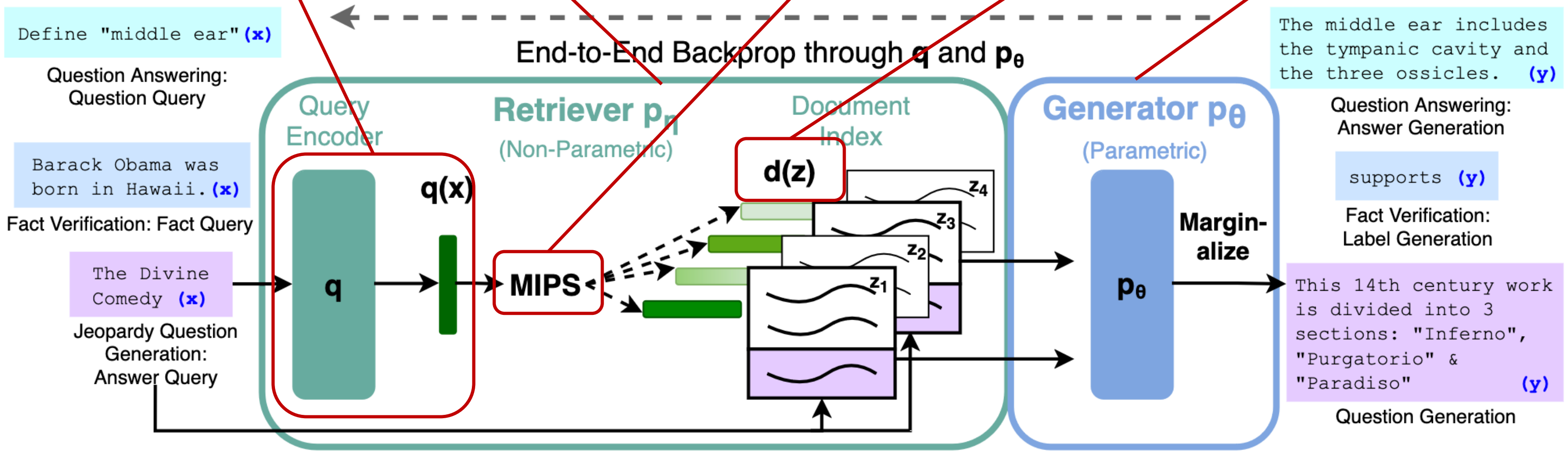
Top-K selection based on Maximum Inner Product Search

BART

$$p_\theta(y_i|x, z, y_{1:i-1})$$

$$\mathbf{q}(x) = \mathrm{BERT}_q(x)$$

$$\mathbf{d}(z) = \mathrm{BERT}_d(z)$$

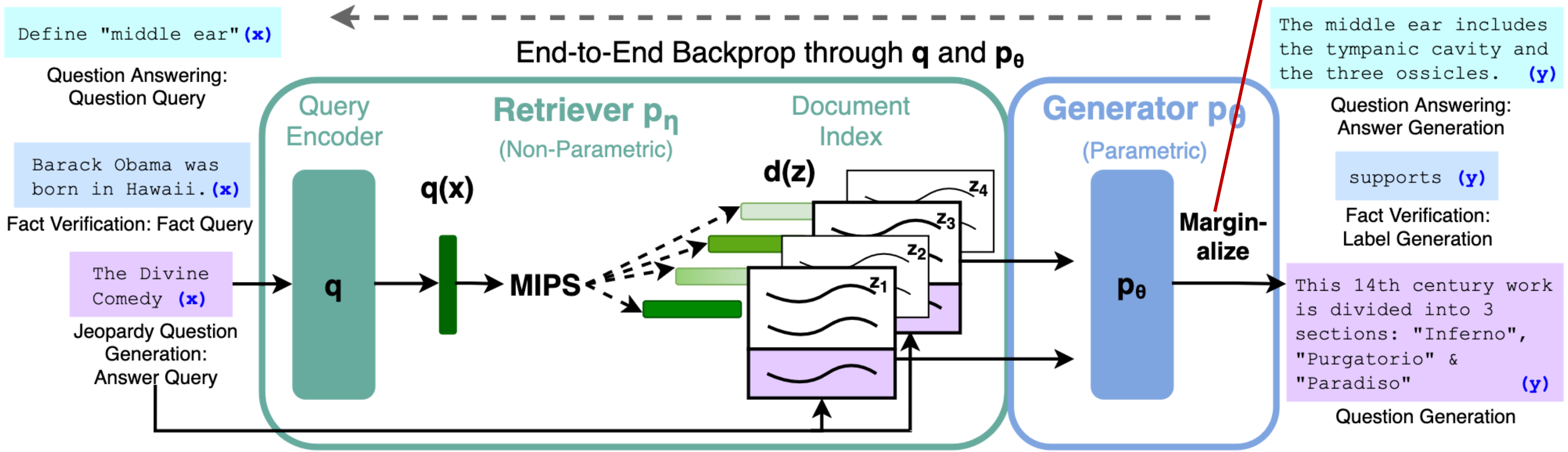


Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, *33*, 9459-9474.

Retrieval Augmented Generation (RAG)

$$p_{\text{RAG-Sequence}}(y|x) \approx \sum_{z \in \text{top-}k(p(\cdot|x))} p_\eta(z|x) p_\theta(y|x,z) = \sum_{z \in \text{top-}k(p(\cdot|x))} p_\eta(z|x) \prod_i^N p_\theta(y_i|x,z,y_{1:i-1})$$

$$\sum_j -\log p(y_j|x_j)$$



Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, *33*, 9459-9474.

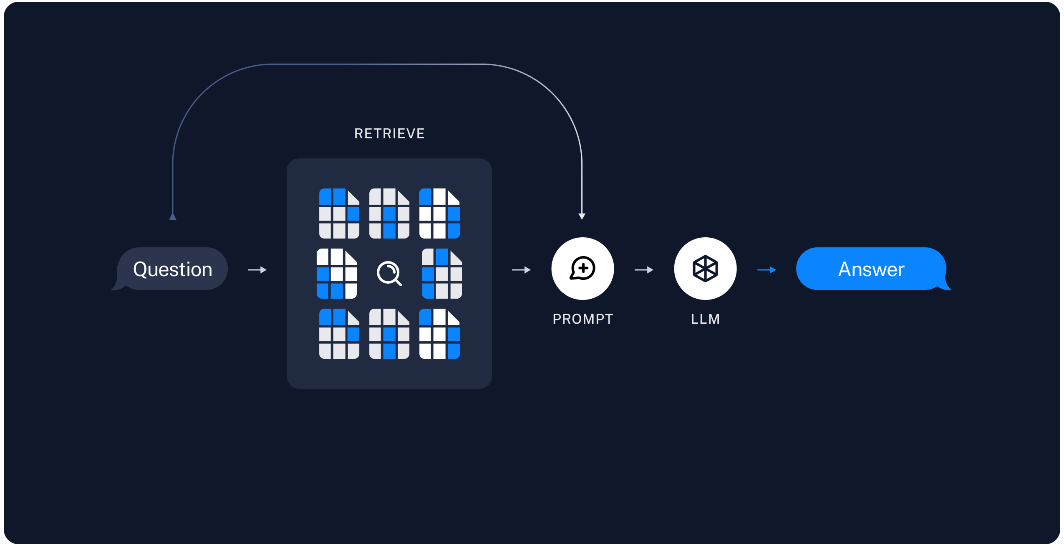
Retrieval Augmented Generation (RAG)

Right now, GPT models are too large for being fine-tuned in a RAG framework

- Vector databases are used in place of ad hoc query encoding
  - Suitable dbs that take into account queries in terms of vector similarity
- In QA, ad hoc frameworks for keeping the history of conversation as a context for the next question

Retrieval Augmented Generation (RAG)

Right now, GPT models are too large for being fine-tuned in a RAG framework



https://ai.meta.com/MediaManagerVideos/videos/244800523626272/