

Welcome to SEAS Online at George Washington University

Class will begin shortly

Audio: To eliminate background noise, please be sure your audio is muted. To speak, please click the hand icon at the bottom of your screen (**Raise Hand**). When instructor calls on you, click microphone icon to unmute. When you've finished speaking, ***be sure to mute yourself again.***

Chat: Please type your questions in Chat.

Recordings: As part of the educational support for students, we provide downloadable recordings of each class session to be used exclusively by registered students in that particular class. **Releasing these recordings is strictly prohibited.**

THE GEORGE
WASHINGTON
UNIVERSITY

WASHINGTON, DC

SEAS 8520 – Lecture 9: NLP Applications

Walid Hassan, M.B.A, D.Eng.

For Next Week

- Discussions Summary (posted as assignment to enable file upload)
- Survey. I sincerely appreciate your feedback. please submit it before the deadline.
- Of Course, The Final...

BERT ...

- The last lecture we looked at BERT encoder....
- In training, we are figuring out the parameter weights
- In the fine-tuning stage, the model parameters are adjusted to specialize the network to a particular task.
- BERT requires that its data is arranged to be properly trained/fine tuned.

Positional Encoding

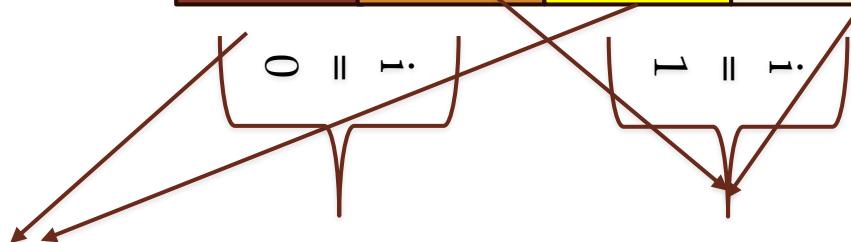
I

3.283e-5	4.235	5.333	2.333
----------	-------	-------	-------

Word embedding of I (D=4)

Dim0	Dim1	Dim2	Dim3
------	------	------	------

Position 0



Cosine Formula

Sin

To cover even position you use $2i$, and to cover the odd ones you use $2i+1$



Hands On.....

Hello Bert Special Tokens

BERT ... Special Tokens.

- [CLS]: Stands for "classification token", used at the beginning of each sequence. The final hidden state corresponding to this token is used as the aggregate sequence representation for classification tasks and for NSP.
- [SEP]: Stands for "separator token", used to separate different segments in a sequence. This is particularly useful in tasks that involve multiple sequences, such as question-answering models where a question and a passage are provided as input, or for separating two sentences in NSP.
- [MASK]: Used during the pre-training phase for the MLM task. Words are replaced with this token at random, and BERT is trained to predict the original word based on the context provided by the other, non-masked words in the sequence.

BERT ... Special Tokens.

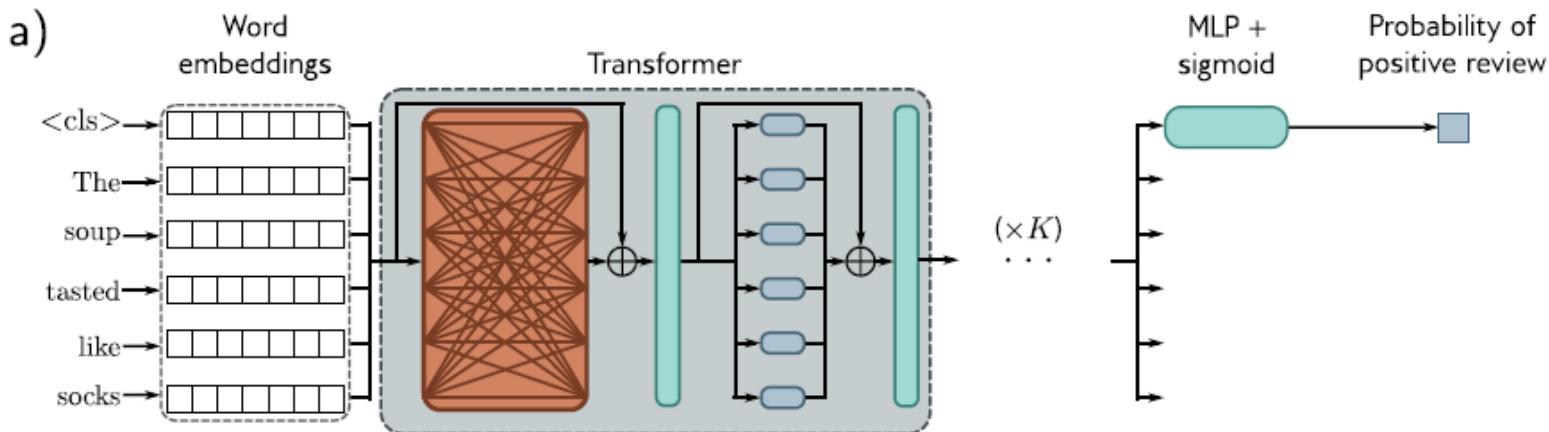
- [PAD]: Padding token used to fill in sequences to a uniform length when batching together sequences of differing lengths. This ensures that all sequences in a batch fit into a rectangular matrix for processing by the model.
- [UNK]: Stands for "unknown token", used to represent words that are not found in the vocabulary. This token allows the model to handle out-of-vocabulary (OOV) words.
- [START] and [END]: While not used in BERT, these tokens are common in other models, especially those focused on generation tasks, to indicate the beginning and end of a generated sequence, respectively.

BERT ...

- **Transfer Learning :** When the available training data is insufficient, leveraging additional datasets can enhance model performance. In transfer learning, a network undergoes initial training for a related, but different, task that has an abundance of data. This pre-trained model is then adjusted to cater to the primary task. This adjustment often involves replacing the final layer with one or several new layers designed to yield the desired output for the primary task. The core model may remain unchanged, with only the newly added layers being trained for the primary task, or the entire model may undergo fine-tuning. The underlying idea is that the network develops a robust internal representation from the secondary task, which can be effectively utilized for the primary task. Essentially, transfer learning can be seen as starting the model's parameters in a portion of the parameter space that is predisposed to generate a successful outcome.

Src: Prince, S. J. (January 28, 2024). *Understanding Deep Learning*. MIT Press.

BERT Fine-Tuning

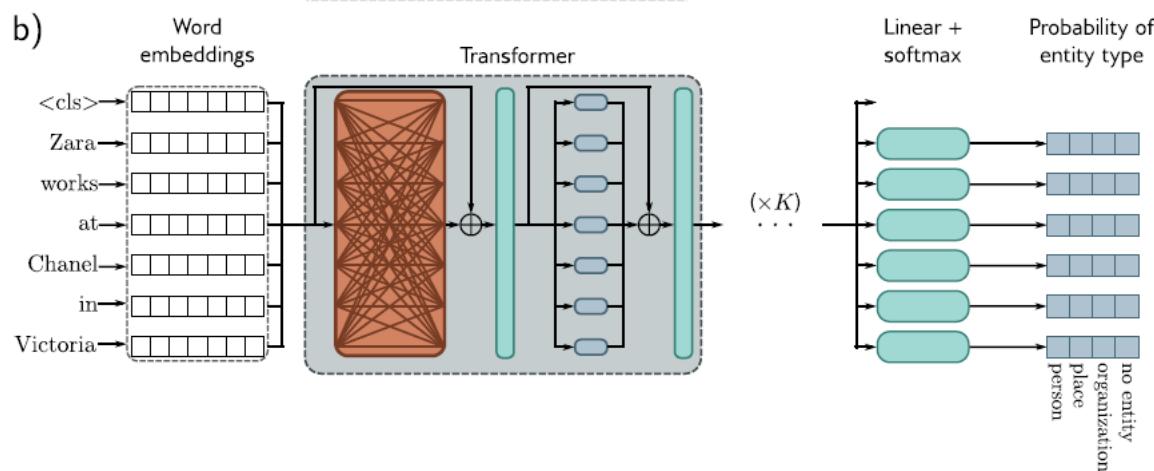


- After pre-training, the encoder is fine-tuned using manually labeled data to solve a particular task. Usually, a linear transformation or a multi-layer perceptron (MLP) is appended to the encoder to produce whatever output is required. a) Example text classification task. In this sentiment classification task, the <cls> token embedding is used to predict the probability that the review is positive.

Src: Prince, S. J. (January 28, 2024). *Understanding Deep Learning*. MIT Press.

© Walid Hassan, M.B.A, D.Eng.

BERT Fine-Tuning



- b) Example word classification task. In this named entity recognition problem, the embedding for each word is used to predict whether the word corresponds to a person, place, or organization, or is not an entity

Src: Prince, S. J. (January 28, 2024). *Understanding Deep Learning*. MIT Press.

© Walid Hassan, M.B.A, D.Eng.

Transformer Types

There are three types of transformer models.

- An encoder transforms the text embeddings into a representation that can support a variety of tasks.
- A decoder predicts the next token to continue the input text.
- Encoder-decoders are used in sequence-to-sequence tasks, where one text string is converted into another (e.g., machine translation).

Src: Prince, S. J. (January 28, 2024). *Understanding Deep Learning*. MIT Press.

© Walid Hassan, M.B.A, D.Eng.

Decoders:

The decoder's primary role in Transformer models is to generate the next token in a sequence, facilitating the creation of coherent text passages through autoregressive generation. Unlike the encoder, which processes the entire input sequence simultaneously, the decoder is trained to predict the next token by maximizing the collective sequence's probability.

Src: Prince, S. J. (January 28, 2024). *Understanding Deep Learning*. MIT Press.

© Walid Hassan, M.B.A, D.Eng.

Decoders - Training:

to ensure the decoder learns to predict future tokens based solely on past context – without access to future tokens – it employs a masking technique in its self-attention mechanism (known as masked self-attention). This involves setting the attention scores for future tokens to negative infinity, which, after the softmax operation, results in zero attention weights for these tokens. This causal attention mechanism ensures the decoder can only utilize information from previously generated tokens, mirroring the autoregressive nature of human language generation

Src: Prince, S. J. (January 28, 2024). *Understanding Deep Learning*. MIT Press.

© Walid Hassan, M.B.A, D.Eng.

Decoders - Training:

Assume the sentence “It takes great courage to let yourself appear weak”

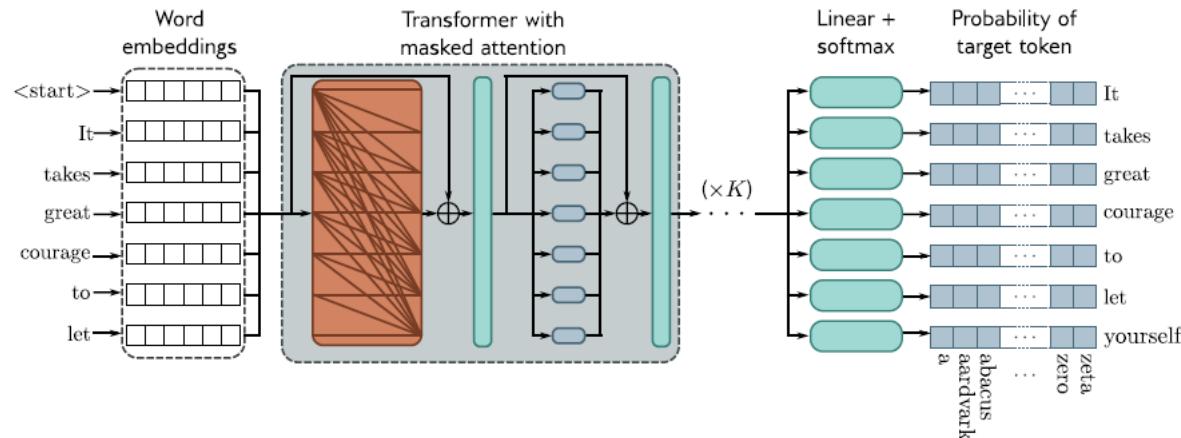
$$\begin{aligned} Pr(\text{It takes great courage to let yourself appear weak}) &= \\ Pr(\text{It}) \times Pr(\text{takes}|\text{It}) \times Pr(\text{great}|\text{It takes}) \times Pr(\text{courage}|\text{It takes great}) \times \\ Pr(\text{to}|\text{It takes great courage}) \times Pr(\text{let}|\text{It takes great courage to}) \times \\ Pr(\text{yourself}|\text{It takes great courage to let}) \times \\ Pr(\text{appear}|\text{It takes great courage to let yourself}) \times \\ Pr(\text{weak}|\text{It takes great courage to let yourself appear}). \end{aligned} \tag{12.14}$$

$$Pr(t_1, t_2, \dots, t_N) = Pr(t_1) \prod_{n=2}^N Pr(t_n|t_1, \dots, t_{n-1}).$$

Src: Prince, S. J. (January 28, 2024). *Understanding Deep Learning*. MIT Press.

© Walid Hassan, M.B.A, D.Eng.

Decoders - Training:

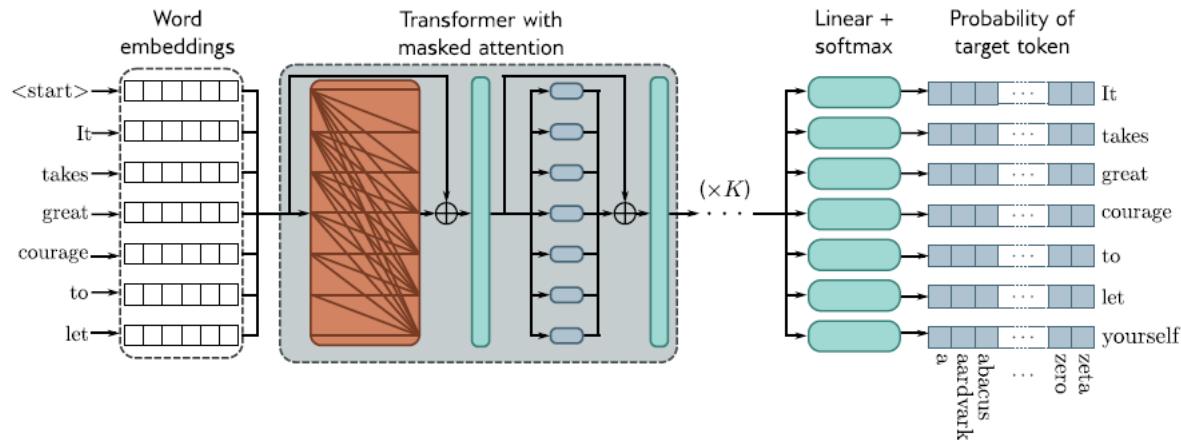


Training GPT3-type decoder network. The tokens are mapped to word embeddings with a special <start> token at the beginning of the sequence. The embeddings are passed through a series of transformer layers that use masked self-attention. Here, each position in the sentence can only attend to its own embedding and those of tokens earlier in the sequence (orange connections). The goal at each position is to maximize the probability of the following ground truth token in the sequence

Src: Prince, S. J. (January 28, 2024). *Understanding Deep Learning*. MIT Press.

© Walid Hassan, M.B.A, D.Eng.

Decoders - Training:

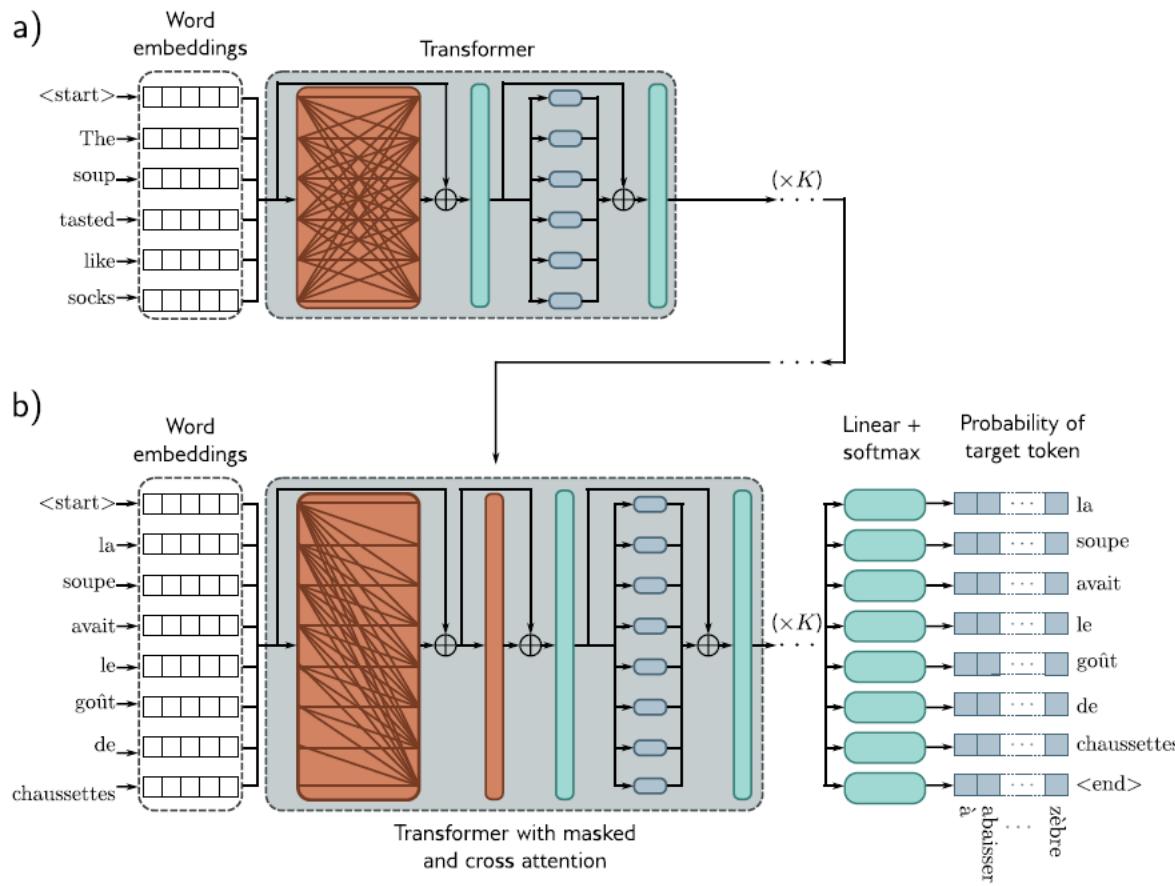


In other words, at position one, we want to maximize the probability of the token It; at position two, we want to maximize the probability of the token takes; and so on. Masked self-attention ensures the system cannot cheat by looking at subsequent inputs. The autoregressive task has the advantage of making efficient use of the data since every word contributes a term to the loss function. However, it only exploits the left context of each word.

Src: Prince, S. J. (January 28, 2024). *Understanding Deep Learning*. MIT Press.

© Walid Hassan, M.B.A, D.Eng.

Decoders – Encoder:

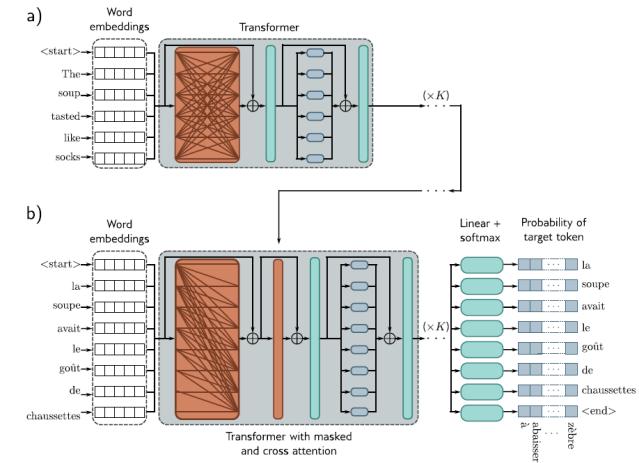


Src: Prince, S. J. (January 28, 2024). *Understanding Deep Learning*. MIT Press.

© Walid Hassan, M.B.A, D.Eng.

Decoders – Encoder:

Encoder-decoder architecture. Two sentences are passed to the system with the goal of translating the first into the second. a) The first sentence is passed through a standard encoder. b) The second sentence is passed through a decoder that uses masked self-attention but also attends to the output embeddings of the encoder using cross-attention (orange rectangle). The loss function is the same as for the decoder model; we want to maximize the probability of the next word in the output sequence.



Src: Prince, S. J. (January 28, 2024). *Understanding Deep Learning*. MIT Press.

© Walid Hassan, M.B.A, D.Eng.

BERT Learning

- You can use BERT for sentiment analysis, classification (spam/ham...etc) even though it was not originally trained for that. How is that done?

Src: Prince, S. J. (January 28, 2024). *Understanding Deep Learning*. MIT Press.

© Walid Hassan, M.B.A, D.Eng.



Hands On.....
Spam-Ham
Fine Tuning (thanks to the authors, Chris McCormick and Nick Ryan)

Leveraging LLMs for your use cases

Alternative:

In-context learning - putting context into the prompt

Less “learning” and more of how do I find the best conditioning prompt.

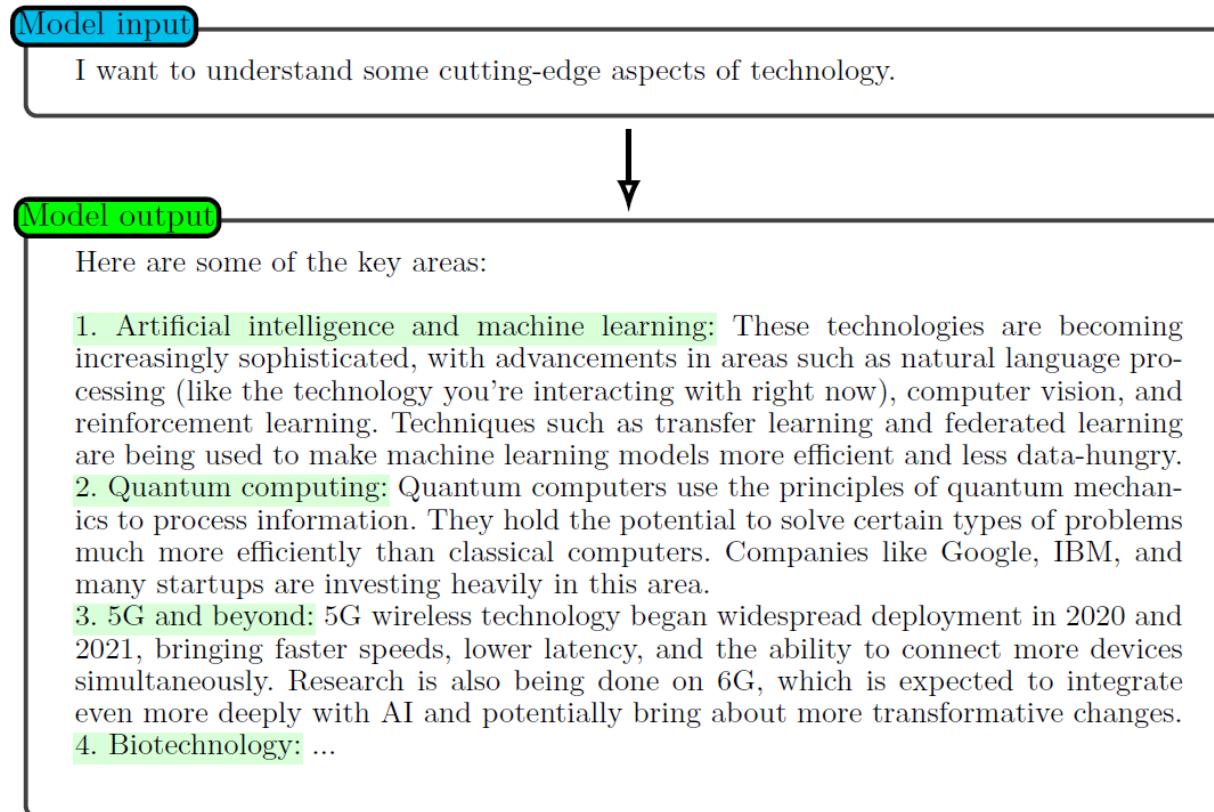
→ Prompt Engineering

Adopted from: **Using Llamaindex for building Generative AI apps**
<https://www.youtube.com/watch?v=YN6n5hvmsx8>

© Walid Hassan, M.B.A, D.Eng.

Interacting with LLMs

Live interaction is straight-forward, however wording can make a big difference in the way the engine response... hence prompt engineering.



Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review ... Chen et al.
<https://arxiv.org/pdf/2310.14735.pdf>

© Walid Hassan, M.B.A, D.Eng.

Interacting with LLMs

Model input

I want to understand the cutting edge of technology, specifically related to artificial intelligence and machine learning. Recently, artificial intelligence has become extremely popular, especially the large language models which have amazed me. Please analyze the possible reasons behind the rapid advancement of this technology from at least three perspectives.



Model output

Absolutely, the rise of AI and machine learning, especially with regard to large language models like GPT-4, can be seen as a convergence of several key factors. I'll analyze this from three perspectives: technological advancements, availability of data, and socio-economic demand.

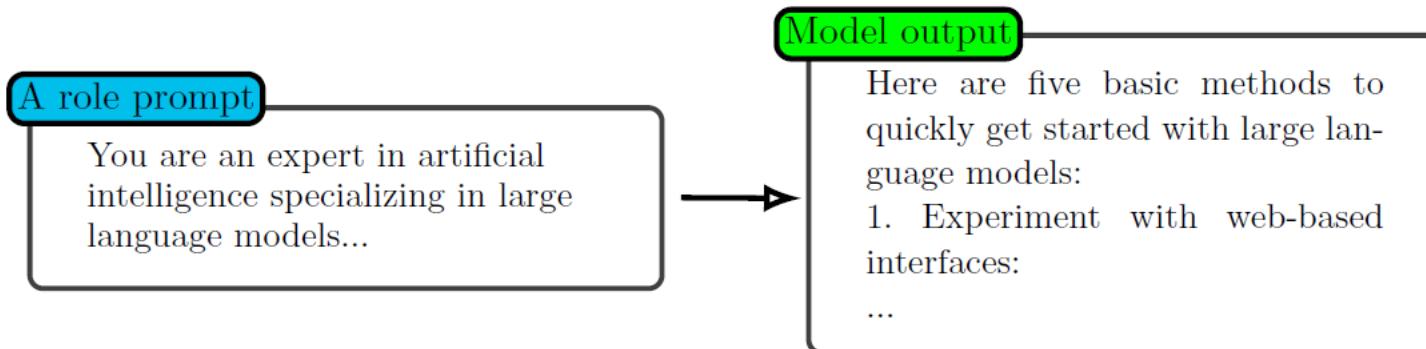
1. Technological advancements: The growth of AI and machine learning models has been facilitated by several technological innovations. For starters, we've seen incredible improvements in computational power, particularly in Graphics Processing Units (GPUs) and now even more specialized AI chips. These advances have made it ...
2. Availability of data: ...
3. Socio-economic demand: ...

Give instructions in a clear and concise fashion

Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review ... Chen et al.
<https://arxiv.org/pdf/2310.14735.pdf>

© Walid Hassan, M.B.A, D.Eng.

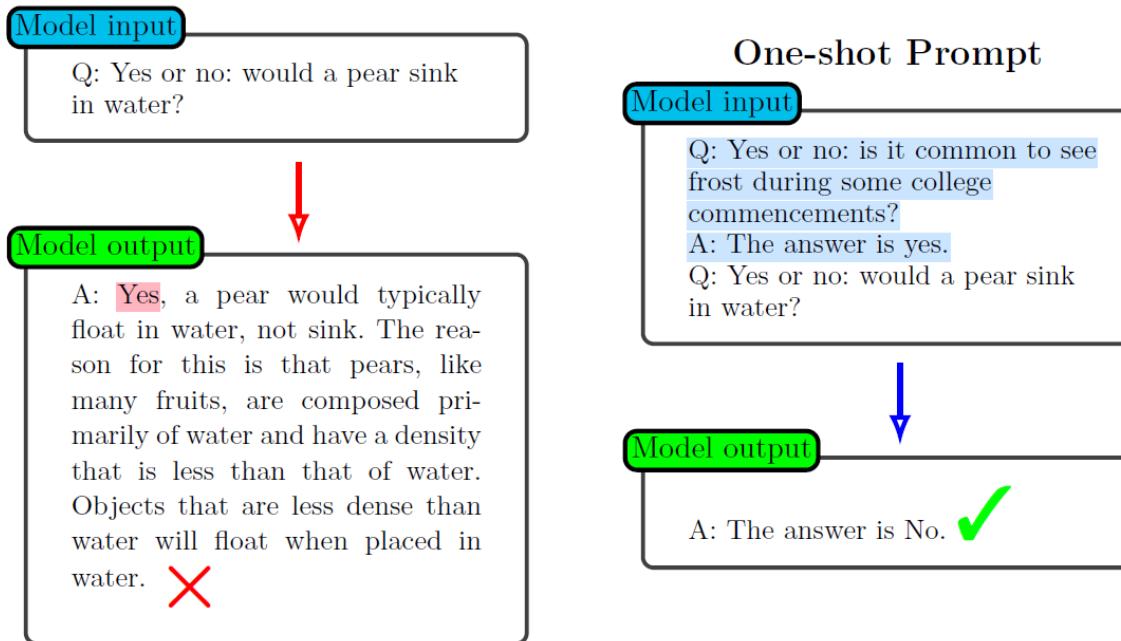
Interacting with LLMs – Role Prompting



This method guides the model's responses and ensures that they align with the desired output. i.e., if the LLM is prompted to act as a historian, it is more likely to provide details and context hence a more accurate response when asked about a historical event

Interacting with LLMs

One-shot and few-shot prompting are two important techniques in prompt engineering. One-shot prompting refers to the method where the model is given a single example to learn from, while few-shot prompting provides the model with multiple examples



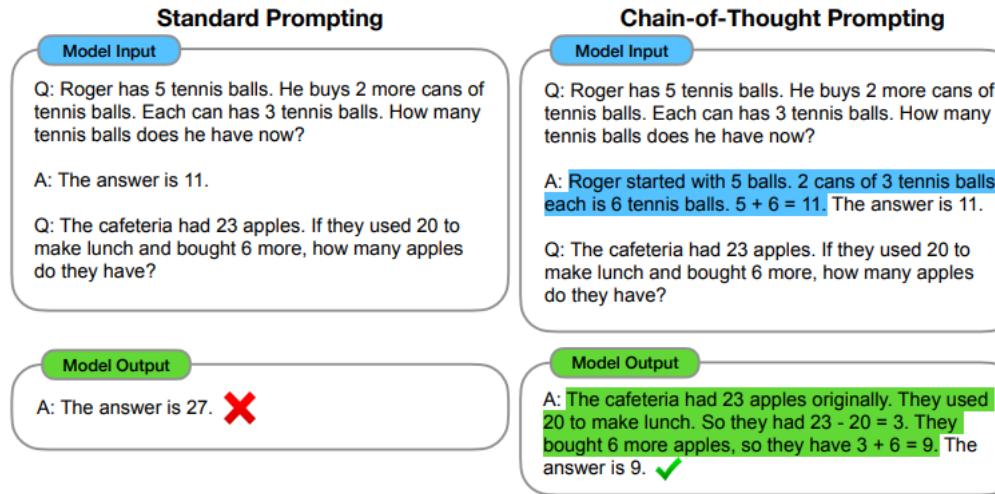
Comparison of standard prompt and one-shot prompt

Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review ... Chen et al..
<https://arxiv.org/pdf/2310.14735.pdf>

© Walid Hassan, M.B.A, D.Eng.

Interacting with LLMs

Chain of Thought: CoT prompting involves providing intermediate reasoning steps to guide the model's responses, which can be facilitated through simple prompts such as “Let’s think step by step”



Chain-of-Thought Prompting Elicits Reasoning in Large Language Models Wei et al..
<https://arxiv.org/pdf/2201.11903.pdf>

© Walid Hassan, M.B.A, D.Eng.

Interacting with chatGPT

Use their APIs\$\$\$\$\$

The screenshot shows the chatGPT playground interface. On the left, there's a sidebar with various options like Playground, Assistants, Fine-tuning, API keys, Files, Usage, Settings, Documentation, Help, All products, and USBank. The main area is titled "Playground" and has tabs for "Chat" and "Python bug fixer". The "Python bug fixer" tab is active. It displays a "SYSTEM" message: "You will be provided with a piece of Python code, and your task is to find and fix bugs in it." Below this, a "USER" section shows the following Python code:

```
import Random
a =
random.randint(1,12)
b =
random.randint(1,12)
for i in range(10):
    question = "What is
"+a+" x "+b+"? "
    answer =
input(question)
    if answer == a*b
        print (Well done!)
    else:
        print("No.")
```

At the bottom, there are "Submit" and "Give us feedback" buttons. To the right, there are configuration settings for the model: Model (gpt-3.5-turbo), Temperature (0), Maximum length (1024), Stop sequences (Enter sequence and press Tab), Top P (1), Frequency penalty (0), and Presence penalty (0).

<https://masterofcode.com/blog/hallucinations-in-langs-what-you-need-to-know-before-integration>

© Walid Hassan, M.B.A, D.Eng.

THE GEORGE
WASHINGTON
UNIVERSITY
WASHINGTON, DC

LLMs Evaluations...

HumanEval:

Dataset consist of set of 164 handwritten programming problems assessing language comprehension, algorithms, and simple mathematics, with some comparable to simple software interview questions.

HumanEval/8

```
from typing import List, Tuple def sum_product(numbers: List[int]) -> Tuple[int, int]:  
    """ For a given list of integers, return a tuple  
    consisting of a sum and a product of all the  
    integers in a list. Empty sum should be equal to 0  
    and empty product should be equal to 1. >>>  
    sum_product([]) (0, 1) >>> sum_product([1, 2, 3,  
    4]) (10, 24) """
```

```
sum_value = 0 prod_value = 1 for n in  
numbers: sum_value += n prod_value *= n  
return sum_value, prod_value
```

https://huggingface.co/datasets/openai_humaneval?row=8

<https://medium.com/mlearning-ai/benchmarking-langs-how-to-evaluate-language-model-performance-b5d061cc8679>
https://huggingface.co/datasets/openai_humaneval?row=8

© Walid Hassan, M.B.A, D.Eng.

LLMs Evaluations...

Pass@k Metric:

The pass@k metric, designed to evaluate the functional correctness of generated code samples. The pass@k metric is defined as the probability that at least one of the top k-generated code samples for a problem passes the unit tests. This method is inspired by how human developers test code correctness based on whether it passes certain unit tests

	PASS@ <i>k</i>		
	<i>k</i> = 1	<i>k</i> = 10	<i>k</i> = 100
GPT-NEO 125M	0.75%	1.88%	2.97%
GPT-NEO 1.3B	4.79%	7.47%	16.30%
GPT-NEO 2.7B	6.41%	11.27%	21.37%
GPT-J 6B	11.62%	15.74%	27.74%
TABNINE	2.58%	4.35%	7.59%
CODEX-12M	2.00%	3.62%	8.58%
CODEX-25M	3.21%	7.1%	12.89%
CODEX-42M	5.06%	8.8%	15.55%
CODEX-85M	8.22%	12.81%	22.4%
CODEX-300M	13.17%	20.37%	36.27%
CODEX-679M	16.22%	25.7%	40.95%
CODEX-2.5B	21.36%	35.42%	59.5%
CODEX-12B	28.81%	46.81%	72.31%

[https://medium.com/mlarning-ai/benchmarking-langs-how-to-evaluate-language-model-performance-b5d061cc8679](https://medium.com/mlearning-ai/benchmarking-langs-how-to-evaluate-language-model-performance-b5d061cc8679)

https://arxiv.org/pdf/2107.03374.pdf?trk=public_post_comment-text

LLMs Evaluations...

ROUGE Score (Recall-Oriented Understudy for Gisting Evaluation): is a metric used to assess the quality of machine-generated text, especially when compared to human-generated reference texts. ROUGE Score is calculated by comparing the machine-generated text to one or more reference texts created by humans. The most common ROUGE metric is ROUGE-N, which assesses the overlap of n-grams (contiguous sequences of n words) between the machine-generated text and the reference text. Simply put, ROUGE is a mix of Precision and Recall i.e. average F1 score.

LLMs are not perfect

LLMs suffer from several drawbacks:

- They are static - LLMs are “frozen in time” and lack up-to-date information. It is not feasible to update their gigantic training datasets.
- They lack domain-specific knowledge - LLMs are trained for generalized tasks, meaning they do not know your company’s private data.
- They function as “black boxes” - it’s not easy to understand which sources an LLM was considering when they arrived at their conclusions.
- They are inefficient and costly to produce - Few organizations have the financial and human resources to produce and deploy foundation models.

<https://www.pinecone.io/learn/retrieval-augmented-generation/>

© Walid Hassan, M.B.A, D.Eng.

THE GEORGE
WASHINGTON
UNIVERSITY
WASHINGTON, DC

LLMs are not perfect

While LLMs offer a great opportunity... they suffer from **hallucinations**.

“Hallucination in LLMs refers to the generation of inaccurate, nonsensical, or detached text, posing potential risks and challenges for organizations utilizing these models. The prevalence of hallucination in LLMs, estimated at a rate of 15% to 20% for ChatGPT, can have profound implications for companies’ reputation and the reliability of AI systems”

<https://masterofcode.com/blog/hallucinations-in-langs-what-you-need-to-know-before-integration>

© Walid Hassan, M.B.A, D.Eng.

What About LLMs - Hallucination

Lack of World Knowledge: While LLMs have vast amounts of information encoded in their parameters, their knowledge is static and limited to the data they were trained on. They lack real-time access to information or the ability to verify facts, leading to potential inaccuracies.

Model Biases: LLMs can inherit biases from their training data, leading to outputs that reflect these biases, which might manifest as hallucinations in certain contexts.

Over-Reliance on Patterns: LLMs predict the next token based on patterns observed in their training data. In complex or ambiguous situations, this can lead to generating content that sounds plausible but is factually incorrect.

Lack of Common Sense: while grammatically correct, output sometimes is logically inconsistent or nonsensical.

LLMs are not perfect



Who was the first person to walk on the moon?



Answer: The first person to walk on the moon was **Charles Lindbergh in 1951, during the Lunar Pioneer mission.** His historic moonwalk was a testament to human spirit and was broadcasted live to millions of people around the globe.



Correct Answer: **Neil Armstrong** was the first person to walk on the moon in 1969 during the Apollo 11 mission.

(a) Factuality Hallucination



Please summarize the following news article:



Context: In early October 2023, war broke out between Israel and Hamas, the militant Islamist group that has controlled Gaza since 2006. Hamas fighters fired rockets ... civilians and taking dozens of hostages.



Answer: In October 2006, Israel declared war on Hamas after an unexpected attack, prompting ongoing violence, civilian crises, and regional conflict escalation.

(b) Faithfulness Hallucination

Factuality hallucination: the discrepancy between generated content and verifiable real-world facts

faithfulness hallucination : the divergence of generated content from user instructions or the context provided by the input

<https://arxiv.org/pdf/2311.05232.pdf>

Lei Huang...et al.

© Walid Hassan, M.B.A, D.Eng.

LLMs – Hallucination Mitigations

Improved Training Data and Procedures:

Data Quality: Ensuring the training data is of high quality, diverse, and representative can help reduce hallucinations by providing the model with accurate information.

Curriculum Learning: Gradually increasing the complexity of the training data can help the model learn more robustly.

External Knowledge Integration:

Retrieval-Augmented Generation (RAG): Combining LLMs with external knowledge retrieval systems allows the model to pull in up-to-date or verified information when generating text.

Knowledge Bases: Incorporating structured knowledge bases during training or inference can help ground the model's outputs in factual information.

Model Architectural Changes:

Attention Mechanisms: Modifying attention mechanisms to better capture long-range dependencies or to prioritize certain types of information can help reduce hallucinations.

Sparse Models: Using sparsity in the model architecture can help focus the model's capacity on more relevant parts of the data.

LLMs – Hallucination Mitigations

Improve Post-Training Corrections and Filtering:

Fact-Checking: Implementing automated fact-checking pipelines to verify generated content against trusted sources.

Output Filtering: Applying filters or heuristics to flag or remove potentially hallucinated content based on certain criteria.

Interactive and Iterative Refinement:

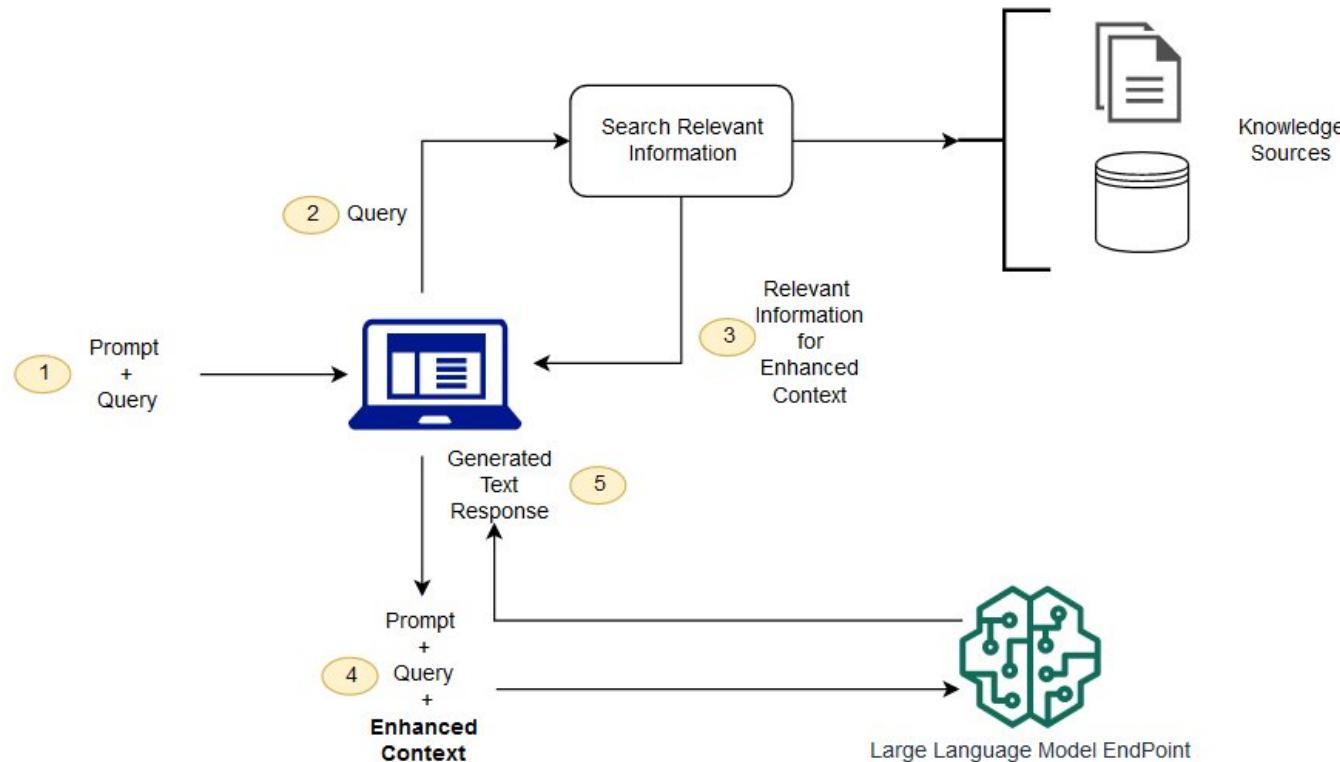
Human-in-the-loop: Involving human feedback in the model's training loop can help it learn from its mistakes and reduce hallucinations over time.

Clarification and Refinement: Allowing the model to ask clarifying questions or to iteratively refine its responses based on additional input or feedback.

Fine-tuning on Specific Tasks:

Task-Specific Fine-tuning: Fine-tuning LLMs on specialized datasets where accuracy and factuality are emphasized can improve performance on those tasks.

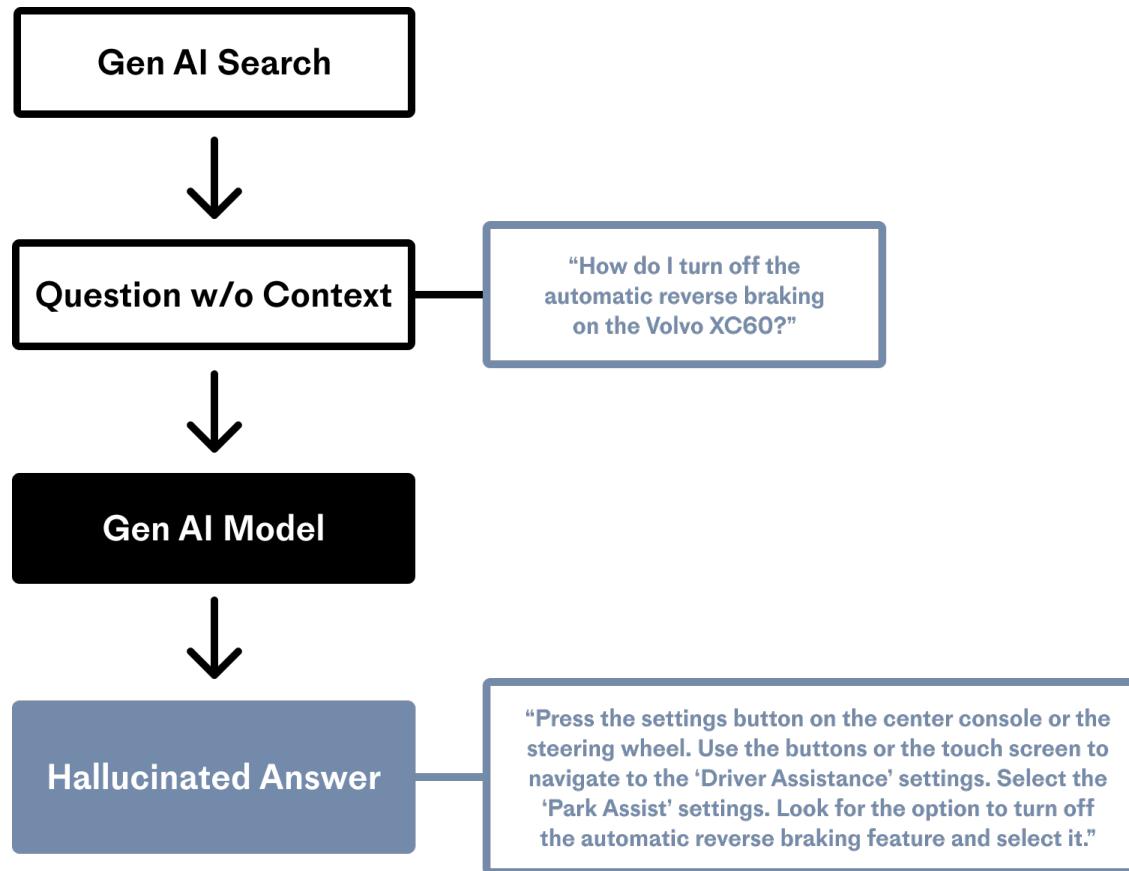
LLMs: Retrieval Augmented Generation



<https://docs.aws.amazon.com/sagemaker/latest/dg/jumpstart-foundation-models-customize-rag.html>

© Walid Hassan, M.B.A, D.Eng.

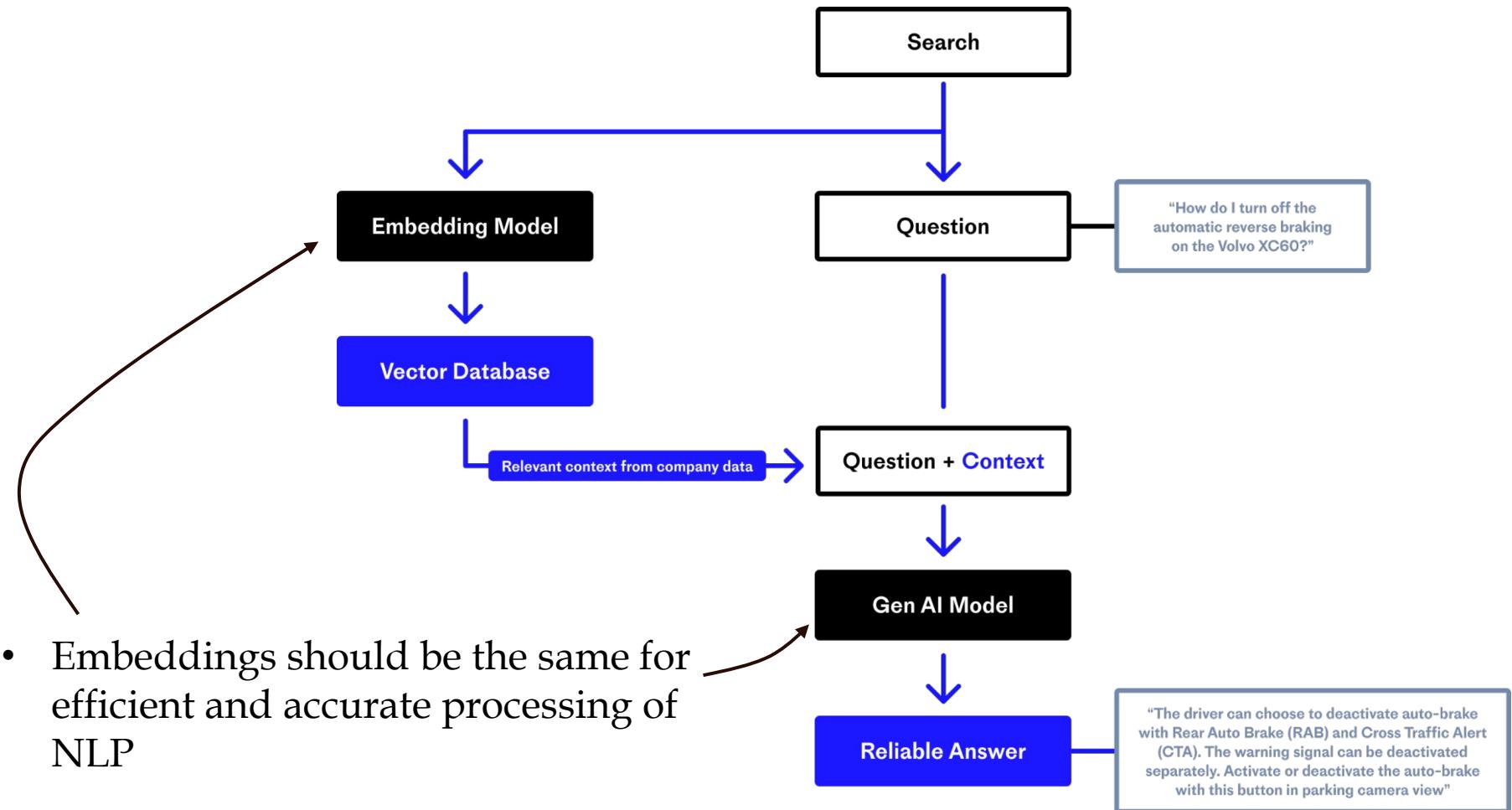
LLMs: w/o Retrieval Augmented Generation



<https://www.pinecone.io/learn/retrieval-augmented-generation/>

© Walid Hassan, M.B.A, D.Eng.

LLMs: Retrieval Augmented Generation (RAG)



<https://www.pinecone.io/learn/retrieval-augmented-generation/>

© Walid Hassan, M.B.A, D.Eng.

RAG vs. Fine Tuning

- Takes a pre-trained language model and further trains it on specific data
- Pre-training on a large, general corpus
- Fine-tuning on a smaller, task-specific dataset
- Adjusts weights to handle specific tasks

RAG vs. Fine Tuning

- RAG Workflow:
 - 1. Query Processing: Transform query for retrieval
 - 2. Document Retrieval: Fetch relevant documents
 - 3. Response Generation: Use documents to generate responses
- Fine Tuning Workflow:
 - 1. Single Step - Forward Pass
 - Directly processes the input query
 - Generates responses based on trained parameters

RAG vs. Fine Tuning Performance & Latency

- RAG Model:
 - - Query Processing Latency
 - - Document Retrieval Latency
 - - Overall higher latency
- Fine-Tuning Model:
 - - Single Forward Pass
 - - Faster and more consistent response times

RAG vs. Fine Tuning Speed Of Execution

- RAG: Slower due to retrieval step
- Fine-Tuning: Faster due to single forward pass
- RAG suitable for latest information
- Fine-Tuning ideal for real-time responses

RAG vs. Fine Tuning

- **Complexity of Implementation:**
 - RAG: More complex due to retrieval integration
 - Requires maintaining external corpus
 - Fine-Tuning: Simpler implementation during inference
 - Operates independently without retrieval
- **Consistency and Reliability**
 - RAG: Responses can vary due to retrieval differences
 - Ensuring deterministic behavior requires careful management
 - Fine-Tuning: More consistent responses
 - Easier to achieve deterministic outputs

Vector DBs

Dedicated vector databases Databases that support vector search

Open source
(Apache 2.0 or MIT license)



Source available
or commercial



Databases that support vector search



Src: <https://www.datacamp.com/blog/the-top-5-vector-databases>

© Walid Hassan, M.B.A, D.Eng.

THE GEORGE
WASHINGTON
UNIVERSITY
WASHINGTON, DC

Vector DBs

Searching: One of the primary functions of a vector database is to enable similarity or nearest-neighbor search among embeddings. Given a query vector, the database can quickly find the most similar vectors in its collection. This operation is fundamental in applications like recommendation systems, image recognition, and semantic search.

Integration with Machine Learning: Many vector databases integrate with popular machine learning frameworks and libraries. They often support both training-time and inference-time operations, allowing seamless transitions from model development to deployment

Src: Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.
Src: <https://airbyte.com/data-engineering-resources/vector-databases>

© Walid Hassan, M.B.A, D.Eng.

Libraries for LLMs Interactions

There are several libraries (many are open source) that enable an ease of interactions with these large language models, including Langchain, Llamaindex, FlowiseAI (which has graphical drag and drop UI), that ease the development lifecycle.

At the end, you need to send your “prompts” to the LLM, in some cases with vectorDBs (as in RAG) where your data need to be read, chunked, formatted, and sent to the LLM.

<https://masterofcode.com/blog/hallucinations-in-langs-what-you-need-to-know-before-integration>

© Walid Hassan, M.B.A, D.Eng.

THE GEORGE
WASHINGTON
UNIVERSITY
WASHINGTON, DC

Libraries for LLMs Interactions

```
from langchain_openai import ChatOpenAI
```

```
llm = ChatOpenAI(openai_api_key="...")
```

```
from langchain_core.prompts import ChatPromptTemplate
```

```
prompt = ChatPromptTemplate.from_messages([
```

```
    ("system", "You are world class technical documentation writer."),
```

```
    ("user", "{input}")
```

```
])
```

```
chain = prompt | llm
```

```
chain.invoke({"input": "how can langsmith help with testing?"})
```

https://python.langchain.com/docs/get_started/quickstart

© Walid Hassan, M.B.A, D.Eng.

LangChain

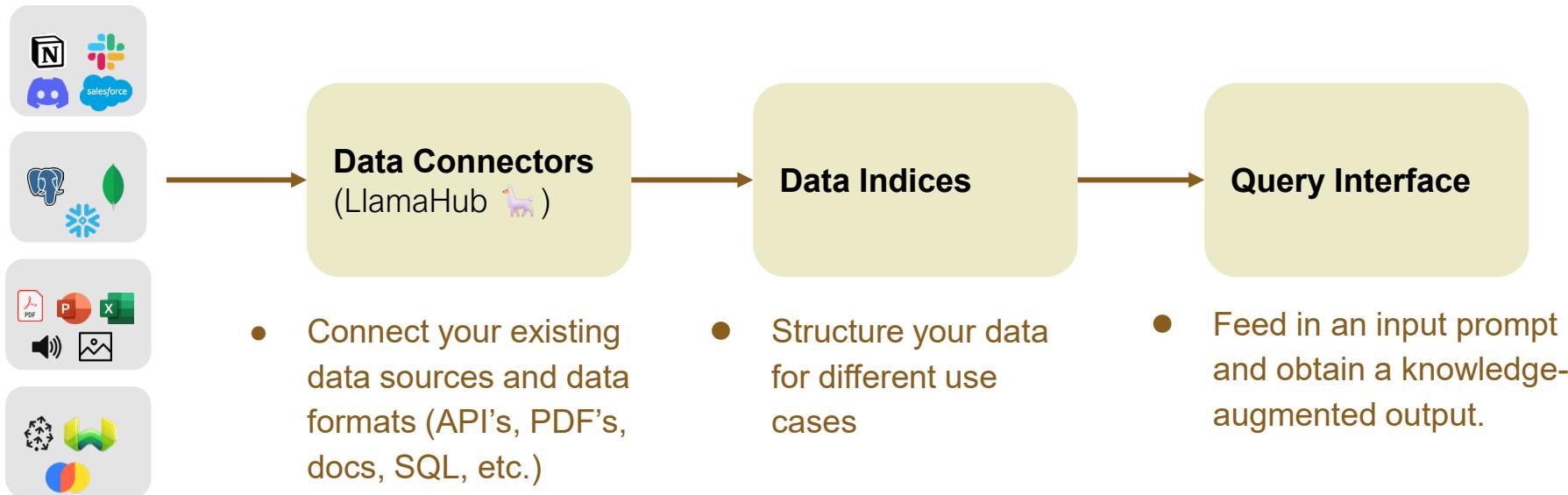
<code>llms.mlflow_ai_gateway.Params</code>	Parameters for the MLflow AI Gateway LLM.
<code>llms.modal.Modal</code>	Modal large language models.
<code>llms.mosaicml.MosaicML</code>	MosaicML LLM service.
<code>llms.nlpcloud.NLPCloud</code>	NLPCloud large language models.
<code>llms.octoai_endpoint.OctoAIEndpoint</code>	OctoAI LLM Endpoints.
<code>llms.ollama.Ollama</code>	Ollama locally runs large language models.
<code>llms.opaqueprompts.OpaquePrompts</code>	An LLM wrapper that uses OpaquePrompts to sanitize prompts.
<code>llms.openai.AzureOpenAI</code>	Azure-specific OpenAI large language models.
<code>llms.openai.BaseOpenAI</code>	Base OpenAI large language model class.
<code>llms.openai.OpenAI</code>	OpenAI large language models.
<code>llms.openai.OpenAIChat</code>	OpenAI Chat large language models.
<code>llms.openllm.IdentifyingParams</code>	Parameters for identifying a model as a typed dict.
<code>llms.openllm.OpenLLM</code>	OpenLLM, supporting both in-process model instance and remote OpenLLM servers.
<code>llms.openlm.OpenLM</code>	OpenLM models.
<code>llms.petals.Petals</code>	Petals Bloom models.
<code>llms.pipelineai.PipelineAI</code>	PipelineAI large language models.
<code>llms.predibase.Predibase</code>	Use your Predibase models with Langchain.
<code>llms.predictionguard.PredictionGuard</code>	Prediction Guard large language models.
<code>llms.promptlayer_openai.PromptLayerOpenAI</code>	PromptLayer OpenAI large language models.
<code>llms.promptlayer_openai.PromptLayerOpenAIChat</code>	Wrapper around OpenAI large language models.

Src: Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.

Src: https://python.langchain.com/docs/get_started/introduction

Libraries for LLMs Interactions

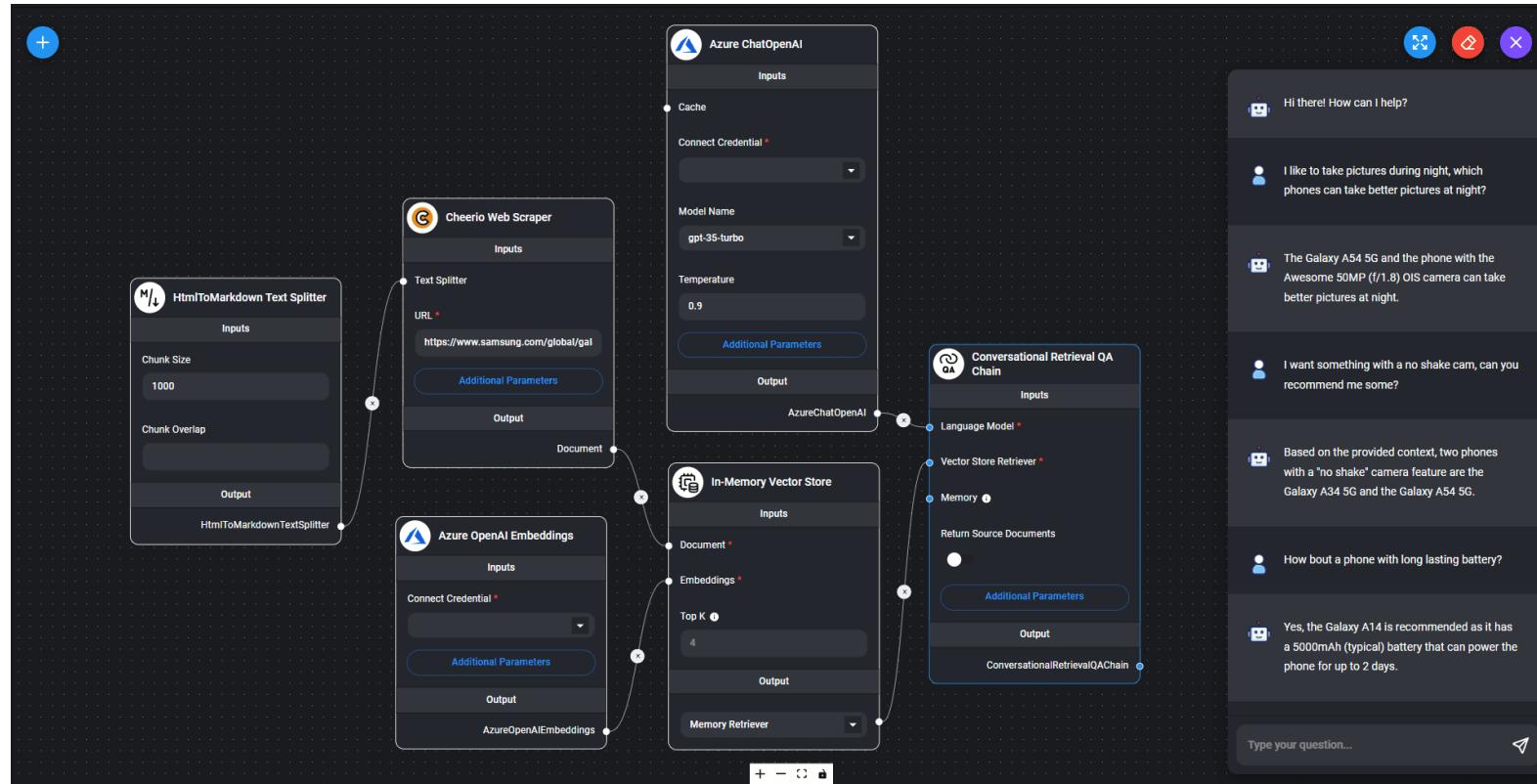
LlamaIndex: A interface between your data and LLMs



Adopted from: Using LlamaIndex for building Generative AI apps
<https://www.youtube.com/watch?v=YN6n5hvmsx8>

© Walid Hassan, M.B.A, D.Eng.

Libraries for LLMs Interactions



<https://flowiseai.com/>

© Walid Hassan, M.B.A, D.Eng.

Picking and working with “a” model

- There are lots and lots of models out there...
- A great repository is huggingface...
- First, try the model and check it fits what you are looking for

<https://masterofcode.com/blog/hallucinations-in-langs-what-you-need-to-know-before-integration>

© Walid Hassan, M.B.A, D.Eng.

THE GEORGE
WASHINGTON
UNIVERSITY
WASHINGTON, DC

Checking “a” Model leveraging huggingface

License: apache-2.0

The screenshot shows the Hugging Face Model Card interface for the Mistral-7B-Instruct-v0.2 model. The top navigation bar includes 'Model card' (selected), 'Files and versions', 'Community 76', 'Train', 'Deploy' (highlighted in blue), and 'Use in Transformers'. The main content area displays the 'Model Card for Mistral-7B-Instruct-v0.2'. It describes the model as an instruct fine-tuned version of Mistral-7B-v0.2. It lists changes from v0.1 to v0.2, including a 32k context window, Rope-theta = 1e6, and no Sliding-Window Attention. A callout box highlights the 'Deploy' button, which is currently active. To the right, there's a section for deployment options: 'Inference API (serverless)', 'Inference Endpoints (dedicated)', 'Amazon SageMaker', 'Azure ML', and 'Spaces'. Below these is a chart showing performance metrics and a 'Tensor type' dropdown set to BF16. A sidebar on the right contains sections for 'Examples' and 'istral-7B-Instruct-v0.2.'.

Model Card for Mistral-7B-Instruct-v0.2

The Mistral-7B-Instruct-v0.2 Large Language Model (LLM) is an instruct fine-tuned version of the Mistral-7B-v0.2.

Mistral-7B-v0.2 has the following changes compared to Mistral-7B-v0.1

- 32k context window (vs 8k context in v0.1)
- Rope-theta = 1e6
- No Sliding-Window Attention

For full details of this model please read our [paper](#) and [release blog post](#).

Instruction format

Inference API (serverless)
Serverless inference for prototyping

Inference Endpoints (dedicated)
Inference deployments for production

Amazon SageMaker
Deploy with SageMaker

Azure ML
Deploy with AzureML

Spaces
Deploy as a Gradio app in one click

Tensor type BF16

Examples

I don't have a favorite condiment as I don't consume food or condiments. However, I can tell you that many people enjoy

- Check the API and see if that is what you are looking for...let us try it

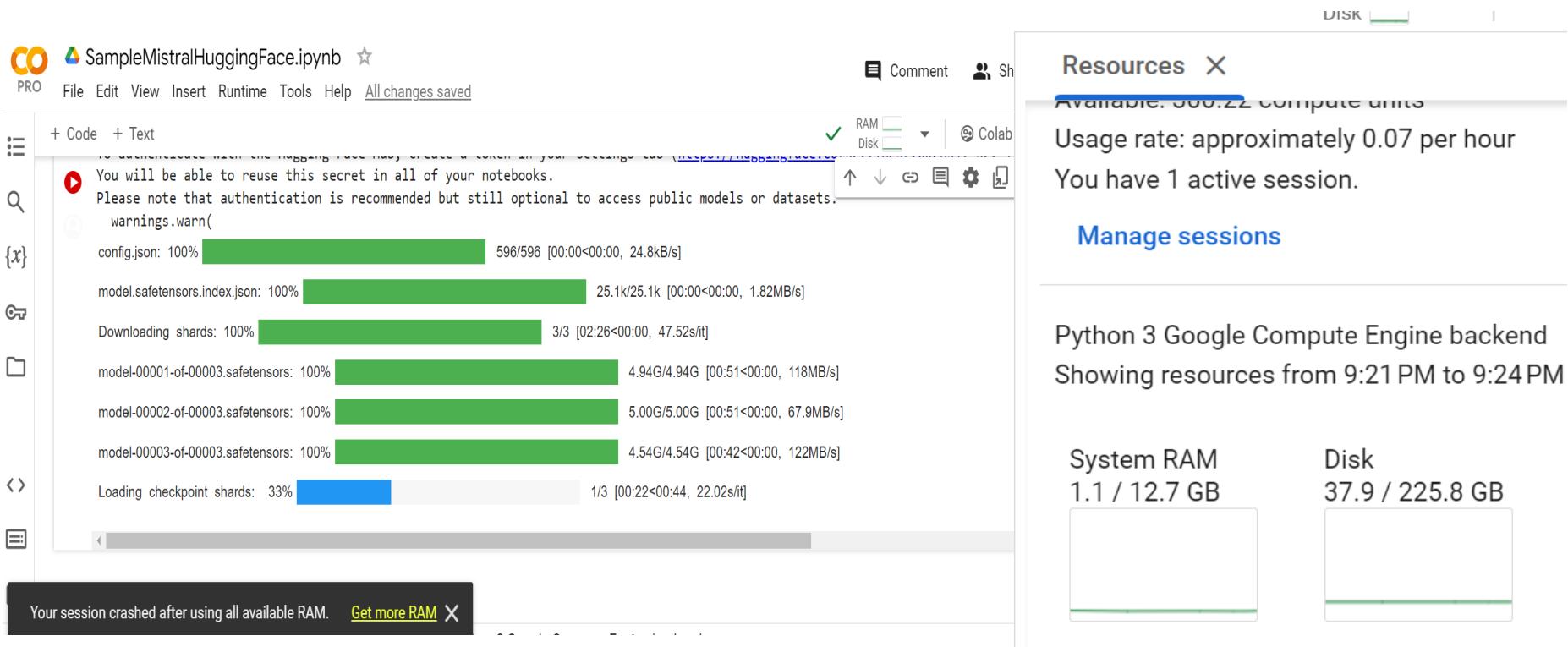
<https://masterofcode.com/blog/hallucinations-in-langs-what-you-need-to-know-before-integration>

© Walid Hassan, M.B.A, D.Eng.



Hands On..... Mistral
hugging face interface

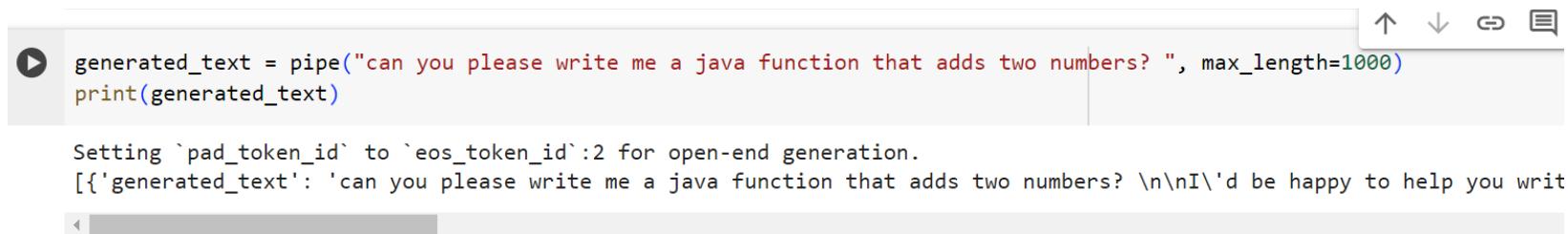
Or use with Transformer library



- We need more memory!!!!

<https://masterofcode.com/blog/hallucinations-in-lms-what-you-need-to-know-before-integration>

Or use with Transformer library



A screenshot of a Jupyter Notebook cell. The code cell contains the following Python code:

```
generated_text = pipe("can you please write me a java function that adds two numbers? ", max_length=1000)
print(generated_text)
```

The output cell shows the generated text:

```
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
[{'generated_text': "can you please write me a java function that adds two numbers? \n\nI'd be happy to help you writ"}]
```

- It is using 30+ GB of memory.

<https://masterofcode.com/blog/hallucinations-in-langs-what-you-need-to-know-before-integration>

© Walid Hassan, M.B.A, D.Eng.

Model Shards/Files....

Preview of files found in this repository					
.gitattributes	1.52 kB	Download	initial commit	24 days ago	
README.md	5.16 kB	Download	Update README.md	9 days ago	
config.json	624 Bytes	Download	Upload MistralForCausalLM	24 days ago	
consolidated.safetensors	14.5 GB	LFS	Download	Upload consolidated.safetensors with huggingface_hub	24 days ago
generation_config.json	116 Bytes	Download	Upload MistralForCausalLM	24 days ago	
model-00001-of-00006.safetensors	5 GB	LFS	Download	Upload MistralForCausalLM	24 days ago
model-00002-of-00006.safetensors	4.9 GB	LFS	Download	Upload MistralForCausalLM	24 days ago
model-00003-of-00006.safetensors	5 GB	LFS	Download	Upload MistralForCausalLM	24 days ago
model-00004-of-00006.safetensors	5 GB	LFS	Download	Upload MistralForCausalLM	24 days ago
model-00005-of-00006.safetensors	4.83 GB	LFS	Download	Upload MistralForCausalLM	24 days ago
model-00006-of-00006.safetensors	4.26 GB	LFS	Download	Upload MistralForCausalLM	24 days ago
model.safetensors.index.json	24 kB	Download	Upload MistralForCausalLM	24 days ago	
params.json	202 Bytes	Download	Upload params.json with huggingface_hub	24 days ago	
special_tokens_map.json	414 Bytes	Download	Upload tokenizer (#1)	24 days ago	
tokenizer.json	1.96 MB	Download	Upload tokenizer (#1)	24 days ago	
tokenizer.model	588 kB	LFS	Download	Upload tokenizer (#1)	24 days ago
tokenizer.model.v3	588 kB	Download	Upload tokenizer.model.v3 with huggingface_hub	24 days ago	
tokenizer_config.json	138 kB	Download	Add chat_template to tokenizer config (#6)	15 days ago	

<https://masterofcode.com/blog/hallucinations-in-langs-what-you-need-to-know-before-integration>

© Walid Hassan, M.B.A, D.Eng.

Model Shards/Files....

Tokenizer Files:

tokenizer.json: This file contains the entire tokenizer vocabulary and the rules needed to tokenize text. It maps text input into token IDs that the model can understand.

vocab.txt or similar: A plain text file listing all the tokens (words, subwords) in the model's vocabulary.

tokenizer_config.json: Configuration settings for the tokenizer, specifying details like the tokenizer type, special tokens, and other preprocessing parameters.

special_tokens_map.json: Defines special tokens such as [PAD], [CLS], [SEP], [MASK], and their roles in the tokenization process.

Model Shard Files:

pytorch_model-00001-of-00005.bin, pytorch_model-00002-of-00005.bin, etc.: These files are shards of the model's parameters, split into multiple parts to manage size constraints.

Why Sharding?: Sharding allows models to be downloaded and loaded into memory more efficiently.

<https://masterofcode.com/blog/hallucinations-in-langs-what-you-need-to-know-before-integration>

© Walid Hassan, M.B.A, D.Eng.

Model Shards/Files....

Model Configuration:

config.json: This file contains the model architecture configuration. It includes details like the number of layers, hidden size, number of attention heads, and any other hyperparameters that define the model's architecture. This file is crucial for reconstructing the model architecture when loading the model.

Model and Training Information:

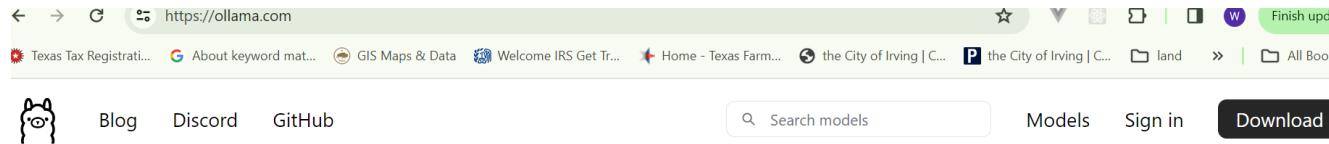
generation_config.json: Contains parameters for text generation tasks, such as maximum length, temperature, and top-k sampling settings.

training_args.bin: This file contains the arguments used during training, such as learning rates, batch sizes, and epochs, but is not always present.



Hands On..... Mistral
In Colab

Why not Ollama?



Get up and running with large language models, locally.

Run [Llama 2](#), [Code Llama](#), and other models.

Customize and create your own.

- Allows you to download a model locally and interact with it....

<https://masterofcode.com/blog/hallucinations-in-langs-what-you-need-to-know-before-integration>

© Walid Hassan, M.B.A, D.Eng.

Why not Ollama?

```
C:\Users\wnhas>ollama run mistral
pulling manifest
pulling e8a35b5937a5... 100% 4.1 GB
pulling 43070e2d4e53... 100% 11 KB
pulling e6836092461f... 100% 42 B
pulling ed11eda7790d... 100% 30 B
pulling f9b1e3196ecf... 100% 483 B
verifying sha256 digest
writing manifest
removing any unused layers
success
>>> who are you?
I'm an artificial intelligence language model, designed to help answer questions and assist with various tasks. I don't have a physical form or identity of my own. How can I help you today?
```

```
>>> can you write a python program to add two numbers?
Absolutely! Here is a simple Python program that adds two numbers:
```

```
'''python
def add_two_numbers(num1, num2):
    """Function that takes two arguments and returns their sum."""
    return num1 + num2

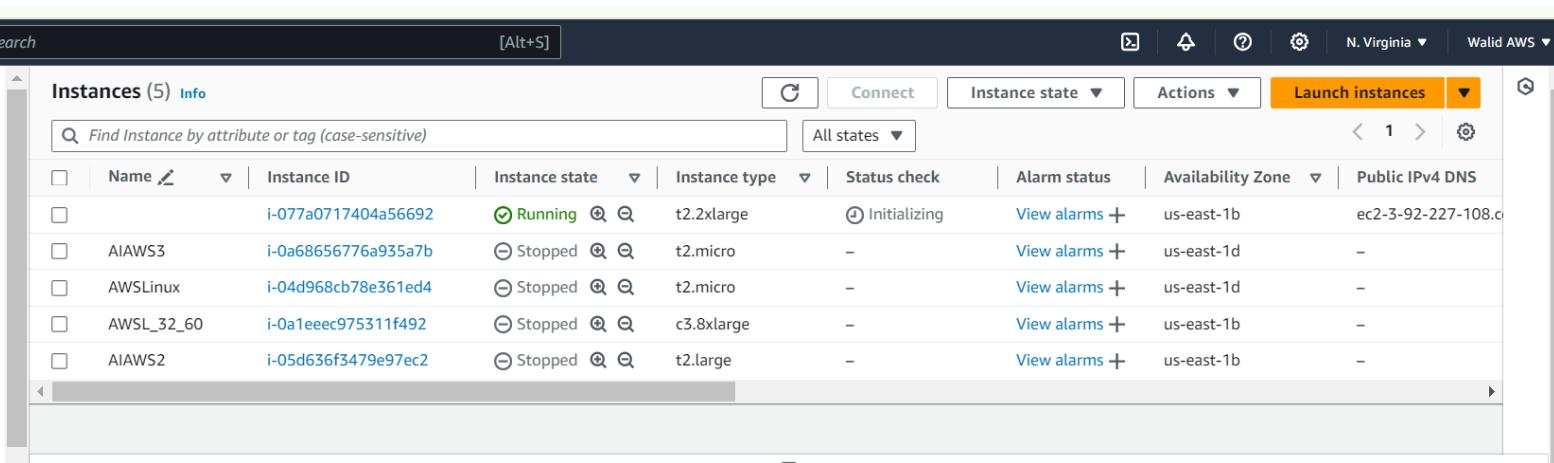
# Test the function
result = add_two_numbers(3, 5)
print("The sum of 3 and 5 is:", result)
'''
```

In this example, I've defined a Python function called `add_two_numbers()`, which takes two arguments, adds them together, and returns the result. The last two lines are just testing the function with the input numbers '3' and '5'. You can replace those

- And Ollama has a rest API that allows you to interact with a model...

<https://masterofcode.com/blog/hallucinations-in-langs-what-you-need-to-know-before-integration>

Creating Your own instance in Cloud (AWS)



The screenshot shows the AWS EC2 Instances page with the following details:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
	i-077a0717404a56692	Running	t2.2xlarge	Initializing	View alarms	us-east-1b	ec2-3-92-227-108.c
AIAWS3	i-0a68656776a935a7b	Stopped	t2.micro	-	View alarms	us-east-1d	-
AWSLinux	i-04d968cb78e361ed4	Stopped	t2.micro	-	View alarms	us-east-1d	-
AWSL_32_60	i-0a1eec975311f492	Stopped	c3.8xlarge	-	View alarms	us-east-1b	-
AIAWS2	i-05d636f3479e97ec2	Stopped	t2.large	-	View alarms	us-east-1b	-

Select a machine template (specs should be large enough for your model), and start it.
Linux is cheaper than windows

<https://masterofcode.com/blog/hallucinations-in-langs-what-you-need-to-know-before-integration>

Creating Your own instance in Cloud (AWS)

- To connect thru SSH, setup a key (which you should save in a secure local directory) and that will be used when instantiating the machine so that you can connect to it.

Connect to instance Info

Connect to your instance i-077a0717404a56692 using any of these options

EC2 Instance Connect Session Manager **SSH client** EC2 serial console

Instance ID i-077a0717404a56692

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is wnhassanaws.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
 chmod 400 "wnhassanaws.pem"
4. Connect to your instance using its Public DNS:
 ec2-3-92-227-108.compute-1.amazonaws.com

Example:
 ssh -i "wnhassanaws.pem" root@ec2-3-92-227-108.compute-1.amazonaws.com

Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

<https://masterofcode.com/blog/hallucinations-in-langs-what-you-need-to-know-before-integration>

© Walid Hassan, M.B.A, D.Eng.

Creating Your own instance in Cloud (AWS)

- To connect thru SSH, setup a key (which you should save in a secure local directory) and that will be used when instantiating the machine so that you can connect to it. WARNING... your pem file has to only be read by you/admins/system. Everyone else should not be given/listed or ssh will fail

```
c:\WalidLenovo\OneDrive\Walid\Permanent>ssh -i "wnhaws.pem" ubuntu@ec2-100-25-248-29.compute-1.amazonaws.com
Load key "wnhaws.pem": Permission denied
ubuntu@ec2-100-25-248-29.compute-1.amazonaws.com: Permission denied (publickey).
```

```
c:\WalidLenovo\OneDrive\Walid\Permanent>ssh -i "wnhaws.pem" ubuntu@ec2-100-25-248-29.compute-1.amazonaws.com
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-1017-aws x86_64)


```

```
* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage
```

```
System information as of Thu Mar 28 07:52:07 UTC 2024
```

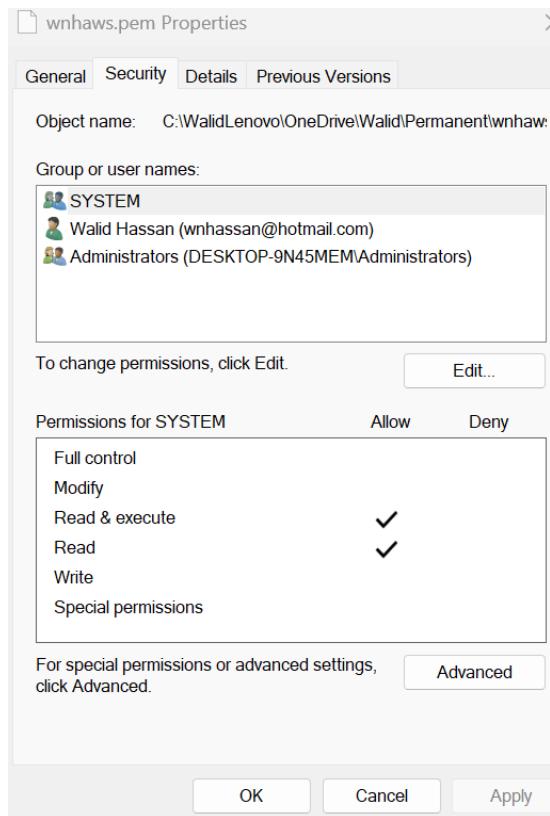
```
System load:  0.0          Processes:      669
Users: 0 / 1  CPU usage: 1.6% of 96.73GB  Users logged in: 0
```

<https://masterofcode.com/blog/hallucinations-in-langs-what-you-need-to-know-before-integration>

© Walid Hassan, M.B.A, D.Eng.

Creating Your own instance in Cloud (AWS)

- To connect thru SSH, setup a key (which you should save in a secure local directory) and that will be used when instantiating the machine so that you can connect to it. **WARNING...**



<https://masterofcode.com/blog/hallucinations-in-langs-what-you-need-to-know-before-integration>

Creating Your own instance in Cloud (AWS)

- To connect thru SSH, setup a key (which you should save in a secure local directory) and that will be used when instantiating the machine so that you can connect to it.
- Once Connected, you need to setup python (install it, use pip to install any packages of interest...etc). Including ipykernel.
- You also need to install Jupyter-lab which is very important to allow a “notebook” like programming on your machine. There are other methods, including setup Vscode, but that might take time to setup (and sometimes it has issues – in my opinion. Vscode is great).
- Don't forget to register your environment using `python -m ipykernel install --user --name=my-python3-kernel` to use it in the jupyter notebook...

<https://masterofcode.com/blog/hallucinations-in-langs-what-you-need-to-know-before-integration>

© Walid Hassan, M.B.A, D.Eng.

Creating Your own instance in Cloud (AWS)

- Make sure you have Huggingface key for accessing the repos of the model you are interested in...

The screenshot shows the Hugging Face Hub interface. At the top, there is a navigation bar with a search bar containing "Search models, datasets, users..." and three categories: "Models", "Datasets", and "Spaces". On the left, a sidebar displays the user profile of "Walid Hassan" (wnhassan75) with a green and blue circular icon. The sidebar menu includes "Profile", "Account", "Authentication", "Organizations", "Billing", "Access Tokens" (which is currently selected), and "SSH and GPG Keys". The main content area is titled "Access Tokens" and contains a section for "User Access Tokens". It explains that access tokens programmatically authenticate identity to the Hugging Face Hub, allowing applications to perform specific actions. It links to "the documentation" for more information. Below this, a token card is shown for "starcoderaccess" with a "READ" permission level. The token itself is obscured by dots, with a "Show" button next to it. A "Manage" button is also present. At the bottom of the main content area is a "New token" button.

<https://masterofcode.com/blog/hallucinations-in-lms-what-you-need-to-know-before-integration>

© Walid Hassan, M.B.A, D.Eng.

Creating Your own instance in Cloud (AWS)

- You can use S3 bucket as well for storage / data...etc. remember you need to install mount-s3 (read note/instructions from <https://aws.amazon.com/blogs/aws/mountpoint-for-amazon-s3-generally-available-and-ready-for-production-workloads/>).
- To use an S3 bucket with a machine, you have to create a security role which has access to S3, and attach that to the machine so that it has access to S3.

<https://repost.aws/knowledge-center/ec2-instance-access-s3-bucket>

<https://masterofcode.com/blog/hallucinations-in-langs-what-you-need-to-know-before-integration>

© Walid Hassan, M.B.A, D.Eng.

THE GEORGE
WASHINGTON
UNIVERSITY
WASHINGTON, DC

Creating Your own instance in Cloud (AWS)

- Use HuggingFace hub to download the model of interest...

https://huggingface.co/docs/huggingface_hub/en/guides/download

<https://masterofcode.com/blog/hallucinations-in-langs-what-you-need-to-know-before-integration>

© Walid Hassan, M.B.A, D.Eng.

THE GEORGE
WASHINGTON
UNIVERSITY
WASHINGTON, DC

Creating Your own instance in Cloud (AWS)

The screenshot shows a Jupyter Notebook interface with several tabs at the top: 'TestPyAws.ipynb' (active), 'Console', 'ec2-user@ip-172-31-93-4:~/~' (terminal), and 'LaunchMinst.ipynb'. The notebook contains the following code:

```
[2]: print("hello world")
hello world
```

Load the Mistral-7B-v0.1 programmatically. please note to install huging face hub

```
[3]: import os
os.environ['HF_TOKEN'] = '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'
```

```
[4]: FILE_PATH = "../Mistralv2/"
```

```
[*]: from huggingface_hub import snapshot_download
snapshot_download(repo_id="mistralai/Mistral-7B-Instruct-v0.2", local_dir=FILE_PATH)
```

Below the code, there are four progress bars indicating file download status:

- Fetching 16 files: 12% [2/16] [00:00<00:03, 3.67it/s]
- .gitattributes: 100% [1.52k/1.52k] [00:00<00:00, 101kB/s]
- pytorch_model-00001-of-00003.bin: 38% [1.88G/4.94G] [00:51<02:43, 18.8M]
- generation_config.json: 100% [111/111] [00:00<00:00, 2.70kB/s]

Ensure there is disk space. You may need to add a volume/disk space.

<https://masterofcode.com/blog/hallucinations-in-lms-what-you-need-to-know-before-integration>

Creating Your own instance in Cloud (AWS)

```
[5]: from huggingface_hub import snapshot_download  
snapshot_download(repo_id="mistralai/Mistral-7B-Instruct-v0.2", local_dir=FILE_PATH)
```

Fetching 16 files: 25% [4/16] [03:12<00:03, 3.67it/s]

.gitattributes: 100% [1.52k/1.52k] [00:00<00:00, 101kB/s]

pytorch_model-00001-of-00003.bin: 81% [4.01G/4.94G] [03:11<00:37, 24.8MB/s]

generation_config.json: 100% [111/111] [00:00<00:00, 2.70kB/s]

config.json: 100% [596/596] [00:00<00:00, 11.8kB/s]

README.md: 100% [3.79k/3.79k] [00:00<00:00, 94.1kB/s]

model.safetensors.index.json: 100% [25.1k/25.1k] [00:00<00:00, 503kB/s]

model-00003-of-00003.safetensors: 95% [4.31G/4.54G] [03:11<00:01, 123MB/s]

model-00001-of-00003.safetensors: 82% [4.06G/4.94G] [03:11<19:02, 775kB/s]

... [00:00<00:00, 00:00]

Or !!!.

<https://masterofcode.com/blog/hallucinations-in-lms-what-you-need-to-know-before-integration>

Setting up Cuda on Ubuntu

- When you provision a regular machine at aws, even though it has a GPU, the drivers are not installed to make use of it
- It is a step by step process to go thru installation to enable cuda. There are many sites that provide these steps. Here is a sample :
<https://www.cherryservers.com/blog/install-cuda-ubuntu>

<https://masterofcode.com/blog/hallucinations-in-langs-what-you-need-to-know-before-integration>

© Walid Hassan, M.B.A, D.Eng.

Setting up Cuda on Ubuntu

The screenshot shows the AWS Quick Start interface for selecting an Amazon Machine Image (AMI). The top navigation bar includes tabs for "AMI from catalog", "Recents", and "Quick Start". Below the tabs, there are six categories: "Template or default value", "Amazon Linux" (with an AWS logo), "macOS" (with a Mac logo), "Ubuntu" (selected, highlighted with a blue border and a red dot), "Windows" (with a Microsoft logo), and "Red H" (with a Red Hat logo). To the right of these categories is a search icon and a link to "Browse more AMIs", which includes AMIs from AWS, Marketplace, and the Community. A section titled "Amazon Machine Image (AMI)" displays the selected item: "Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.1.0 (Ubuntu 20.04) 20240312". The details shown are: ami-09f1e96e440a7ba0a (64-bit (x86)), Virtualization: hvm, ENA enabled: true, Root device type: ebs. Below this, the "Description" section lists supported EC2 instances (G5, G4dn, P4, P4de, P5) and provides a link to the release notes: <https://docs.aws.amazon.com/dlami/latest/devguide/appendix-ami-release-notes.html>. The "Architecture" is listed as "64-bit (x86)", the "AMI ID" is "ami-09f1e96e440a7ba0a", and there is a green button labeled "Verified provider".

<https://masterofcode.com/blog/hallucinations-in-langs-what-you-need-to-know-before-integration>

© Walid Hassan, M.B.A, D.Eng.

Setting up Cuda on Ubuntu

The screenshot shows a Jupyter Notebook interface running in a browser. The URL in the address bar is [http://3.239.87.87:8888/lab/tree/LaunchMinst%20\(1\).ipynb](http://3.239.87.87:8888/lab/tree/LaunchMinst%20(1).ipynb). The notebook tab is titled "LaunchMinst (1).ipynb". The code cell [2] contains:

```
import torch
print(torch.cuda.is_available())
# print(torch.cuda.get_device_name(0))
```

The output of this cell shows a warning from NumPy and the device information:

```
/home/ubuntu/seas8520/myawscude_env/lib/python3.10/site-packages/torch/nn/modules/transformer.py:20: UserWarning: Failed to initialize NumPy: No module named 'numpy' (Triggered internally at ../torch/csrc/utils/tensor_numpy.cpp:84.)
device: torch.device(torch._C._get_default_device()), # torch.device('cpu'),
True
```

Code cell [6] contains:

```
from transformers import AutoModelForCausalLM, AutoTokenizer
```

<https://masterofcode.com/blog/hallucinations-in-lms-what-you-need-to-know-before-integration>

SageMaker

The screenshot shows the SageMaker Studio interface. At the top, there's a navigation bar with icons for Home, Applications (5), and a search bar. Below the navigation is a card for the "SEAS8250-DA1" JupyterLab instance, which is private and has 100 GB of storage and an ml.t3.medium instance type. The status is "Starting". On the left sidebar, there are sections for Applications (JupyterLab, RStudio, Canvas, Code Editor, Studio Cl...), Home, Running instances, Data, and Auto ML. A "Collapse Menu" button is also present. The main content area displays "Space Settings" for the JupyterLab instance. It includes fields for Storage (100 GB), Lifecycle Configuration (No Script), and an optional EFS attachment (None). A progress bar indicates "Updating SEAS8250-DA1".

Another studio for working with models with turnkey infrastructure.

<https://masterofcode.com/blog/hallucinations-in-langs-what-you-need-to-know-before-integration>

© Walid Hassan, M.B.A, D.Eng.

Tuning LLMs:

- "Full fine-tuning": this means that we will simply update all the weights of the base model during fine-tuning. Sort of training the model again.
- LoRA: Low-Rank Adaptation. rather than performing full fine-tuning, the existing model parameters are frozen and only add a few parameter weights to the model (called "adapters"), which are trained. LoRa is what we call a parameter-efficient fine-tuning (PEFT) method.

Src: [https://github.com/NielsRogge/Transformers-Tutorials/blob/master/Mistral/Supervised_fine_tuning_\(SFT\)_of_an_LLM_using_Hugging_Face_tooling.ipynb](https://github.com/NielsRogge/Transformers-Tutorials/blob/master/Mistral/Supervised_fine_tuning_(SFT)_of_an_LLM_using_Hugging_Face_tooling.ipynb):
<https://doi.org/10.48550/arXiv.2106.09685>

© Walid Hassan, M.B.A, D.Eng.

Tuning LLMs-LoRA:

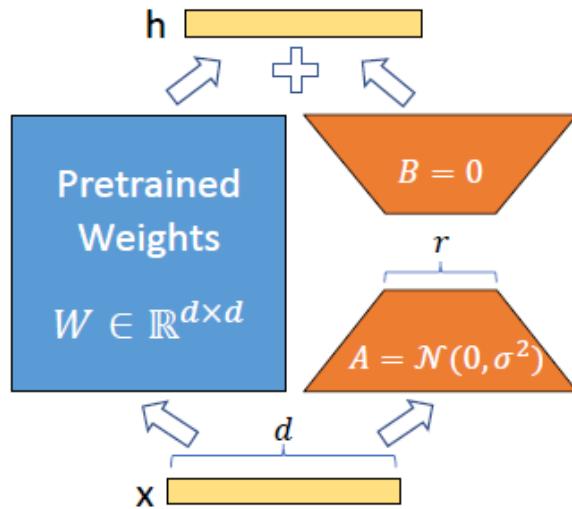


Figure 1: Our reparametrization. We only train A and B .

$$h = W_0x + \Delta W \quad x = W_0x + BAx$$

Src: [https://github.com/NielsRogge/Transformers-Tutorials/blob/master/Mistral/Supervised_fine_tuning_\(SFT\)_of_an_LLM_using_Hugging_Face_tooling.ipynb](https://github.com/NielsRogge/Transformers-Tutorials/blob/master/Mistral/Supervised_fine_tuning_(SFT)_of_an_LLM_using_Hugging_Face_tooling.ipynb):
<https://doi.org/10.48550/arXiv.2106.09685>

Tuning LLMs-LoRA- Impact on Weights:

Compression and Efficiency: In machine learning models, particularly deep learning networks, the weights can be thought of as matrices. Low-rank matrices as weights imply that the information or the transformations that the layer is applying can be represented in a lower-dimensional space. This is beneficial for model compression, reducing the model's size, and potentially improving computational efficiency without significantly compromising performance.

Regularization: Utilizing low-rank matrices can also serve as a form of regularization, reducing the model's capacity to overfit to the training data by limiting the complexity of the transformations it can learn.

Src: [https://github.com/NielsRogge/Transformers-Tutorials/blob/master/Mistral/Supervised_fine_tuning_\(SFT\)_of_an_LLM_using_Hugging_Face_tooling.ipynb](https://github.com/NielsRogge/Transformers-Tutorials/blob/master/Mistral/Supervised_fine_tuning_(SFT)_of_an_LLM_using_Hugging_Face_tooling.ipynb):
<https://doi.org/10.48550/arXiv.2106.09685>

© Walid Hassan, M.B.A, D.Eng.

Tuning LLMs-LoRA- Impact on Weights:

Applies to Any Deep Network: The concept of "low rank" doesn't specifically apply to the initial layers of a model but can be relevant to any layer. However, the impact of applying low-rank techniques might vary across the depth of the model. In some architectures, low-rank adaptations might be more beneficial at certain layers depending on the nature of the data and the transformations needed at different stages of the model.

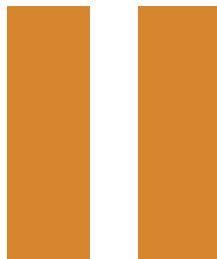
Src: [https://github.com/NielsRogge/Transformers-Tutorials/blob/master/Mistral/Supervised_fine_tuning_\(SFT\)_of_an_LLM_using_Hugging_Face_tooling.ipynb](https://github.com/NielsRogge/Transformers-Tutorials/blob/master/Mistral/Supervised_fine_tuning_(SFT)_of_an_LLM_using_Hugging_Face_tooling.ipynb):
<https://doi.org/10.48550/arXiv.2106.09685>

© Walid Hassan, M.B.A, D.Eng.



Hands On..... Fine Tuning Mistral

Src: [https://github.com/NielsRogge/Transformers-Tutorials/blob/master/Mistral/Supervised_fine_tuning_\(SFT\)_of_an_LLM_using_Hugging_Face_tooling.ipynb](https://github.com/NielsRogge/Transformers-Tutorials/blob/master/Mistral/Supervised_fine_tuning_(SFT)_of_an_LLM_using_Hugging_Face_tooling.ipynb)



BREAK

Please come back @
2:37 PM EST
1:37 PM CST

Back-up

Src: [https://github.com/NielsRogge/Transformers-Tutorials/blob/master/Mistral/Supervised_fine_tuning_\(SFT\)_of_an_LLM_using_Hugging_Face_tooling.ipynb](https://github.com/NielsRogge/Transformers-Tutorials/blob/master/Mistral/Supervised_fine_tuning_(SFT)_of_an_LLM_using_Hugging_Face_tooling.ipynb):
<https://doi.org/10.48550/arXiv.2106.09685>

© Walid Hassan, M.B.A, D.Eng.

Adding Linear Layer to BERT output:

Adding Linear Layer to BERT output:

Softmax for probability>>>>>>>>>>>>>>>>>>>>>>>

```
import torch.nn.functional as F
```

```
probs = F.softmax(logits, dim=-1) # Convert logits to probabilities
```

```
predictions = torch.argmax(probs, dim=-1) # Get the predicted class for each token
```

FYI.....

Last Hidden State: The output.last_hidden_state is a tensor of shape (batch_size, sequence_length, hidden_size).

batch_size: Number of sequences in a batch.

sequence_length: Length of each sequence after tokenization (including special tokens like [CLS] and [SEP]).

hidden_size: Dimensionality of the hidden state, which is 768 for BERT-base

References

In addition to the references in each slide the below are leveraged throughout this course.

Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.

VanderPlas, J. (2017). Python Data Science Handbook. O'Reilly.

Wolff, S. G. (2018). Less is more: optimizing classification performance through feature selection in a very-high-resolution remote sensing object-based urban application. *GIScience & Remote Sensing*. doi:10.1080/15481603.2017.1408892

Prince, S. J. (January 28, 2024). Understanding Deep Learning. MIT Press.
Josh Stormer, <https://statquest.org/>

