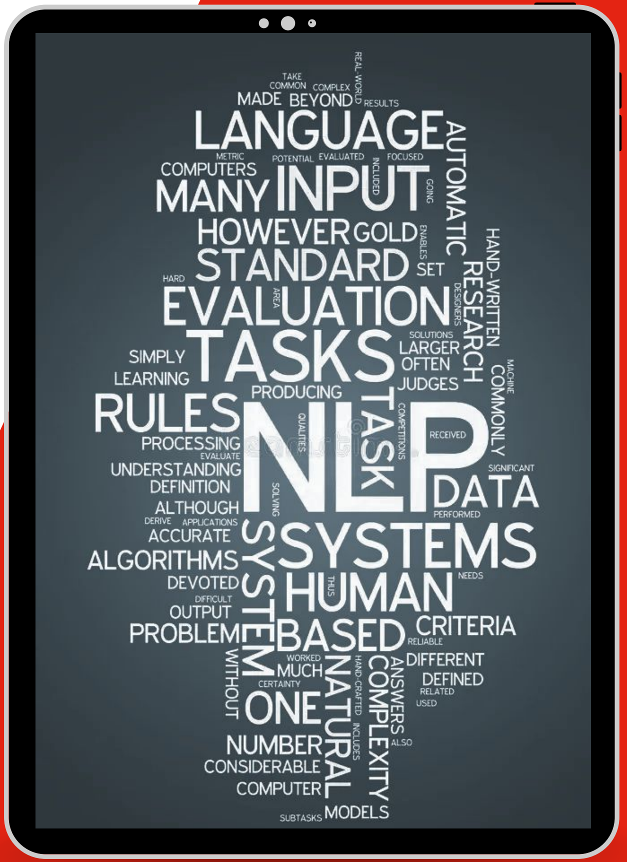
LLM

Muhajir Akbar Hasibuan

Class Meeting 11

2023



Muhajir Akbar Hasibuan
Data Scientist

muhajirakbarhsb@gmail.com
+62 85297023234
Jakarta, Indonesia
linkedin.com/in/muhajirakbarhsb
github.com/muhajirakbarhsb
medium.com/@muhajirakbarhsb

Experience

Telkom Indonesia *Sept 2020 - present*

- Develop internal and external use cases
- Provide data understanding in making a model
- Provide Preparation and data engineering according to the use case implemented
- Provide data validation so that the analysis results are as expected
- Building modeling for the development or improvement of internal and external programs
- Provide descriptive and diagnostic insight into data processing
- Recommend and define new growth hacking strategy for digital marketing team

Project

- Pioneered a Robust Big Data Solution for MyIndihome, Revolutionizing Customer Experiences
- Orchestrated a High-Performing ML Team, Elevating myIndihome TV's Personalized Content Impact by 25%
- Envisioned and Executed a Cutting-Edge Big Data Solution for Elevated Customer Engagement on myIndihomeTV
- Fueled Business Insights via Dynamic Data Profiling, Performance Dashboards, and Insights for Langit Musik, RBT, and Upoint
- Engineered Innovative Big Data Solutions that Propelled Growth for PadiUmkm (E-Commerce)
- Powered Success for GameQoo through Strategic Big Data Solutions
- Architected and Established MLOps Framework, Elevating Telkom Indonesia's Digital Business Products
- Crafted Visionary Video Analytics Solutions for Telkom Indonesia's Revolutionary Digital Business IoT Product

MEMBER OF DATA SCIENTIST TASK FORCE | NOVEMBER 2021 - PRESENT
MEMBER OF AI TASK FORCE | NOVEMBER 2022- PRESENT

A Pool of data scientists and AI Engineer Expert in Telkom Indonesia. It was established to leverage data-driven culture for decision-making within the organization. (Applied Research, Standardization, Consultation)

Achievement

BEST Talent of the Year at Telkom Indonesia, Digital Business and Technology Division - Digital Technology and Platform -2022

Codex by Telkom Indonesia *Apr 2020- Sept 2020*

- Building Data Pipeline for Langit Musik Recommender System
- Business Analytic for Langit Musik
- Build Recommender System for Langit Musik

Universitas Syiah Kuala *2015-2020*

- Bachelor of Science, Statistics

Skills

Hard Skills

- Data Analytics
- Statistics
- Machine Learning
- Deep Learning
- MlOps
- Business Intelligence
- Data Engineering
- Cloud Computing
- Time Series & Demand Forecasting
- Natural Language Processing
- Fraud & Anomaly Detection
- Computer Vision

Tools Skills

- Python
- Pyspark
- Sql
- Apache Airflow
- GCP
- Docker
- MLflow
- Prometheus
- Evidently
- Grafana
- Redash, Metabase, Superset, Looker Studio
- Pytorch

Class Meeting 1: Introduction to NLP (1 session)
Introduction to NLP
Basics of text data and its characteristics
The importance of NLP in today's world
Overview of the course structure and objectives

Class Meeting 2: Text Preprocessing (1 session)
Understanding the text preprocessing pipeline
Tokenization, stemming, and lemmatization
Stop words removal
Hands-on exercises with Python for text preprocessing

Class Meeting 3: Text Representation (1 session)
Introduction to text representation techniques
Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF)
n-gram
Word embeddings (Word2Vec, GloVe)**just an introduction
Practical exercises on text representation

Class Meeting 4: Introduction to ML, DL Supervised and Unsupervised (1 session)
Introduction to deep learning (ML)
Introduction to deep learning (DL)
DL vs. traditional machine learning (ML)
Machine Learning and their applications
Neural networks and their applications
Basic ML DL concepts and terminology

Class Meeting 5: Sentiment Analysis (1 session)
What is sentiment analysis?
Data collection and labeling for sentiment analysis
Building a sentiment analysis model
Practical sentiment analysis examples

Class Meeting 6: Text Classification (1 session)
Introduction to text classification
Binary and multi-class classification
Building a text classification model
Real-world text classification examples

Class Meeting 6 Introduction to Topic Modeling(1 session)
Introduction to topic Modelingg
Topic Modeling Implementation
LDA(Latent dirichlet allocation)
Hands-on exercises with Topic Modeling

mid term

Class Meeting 8: **Sesi Khusus deep Learning**

Class Meeting 9: Introduction to Word Embeddings (1 session)
Fundamentals of word embeddings
Word2Vec, GloVe, and FastText
Word embedding applications
Hands-on exercises with word embedding

Class Meeting 10: Attention Mechanisms (1 session)
Introduction to attention mechanisms
Self-attention and multi-head attention
Transformers architecture
Practical examples of attention in NLP

Class Meeting 11: Transformer Models (1 session)
In-depth study of the Transformer model
Pre-trained transformer models (BERT, GPT-2)
Fine-tuning transformers for NLP tasks
Transformer-based applications

Class Meeting 12: Advanced NLP Topics (1 session)
Advanced NLP topics such as BERT, XLNet, and RoBERTa
Transfer learning in NLP
Ethics in NLP
Recent developments and trends in NLP

Class Meeting 13: Advanced NLP Topics and Deployment Process in Industry (1 session)
Introduction to LLM
Introduction how industry utilize NLP to generate revenue
Introduction MLOps for NLP(bonuses from practitioners)

Class Meeting 14-15: Project Work (2 sessions)
Dedicated sessions for students to work on NLP projects with guidance and assistance.

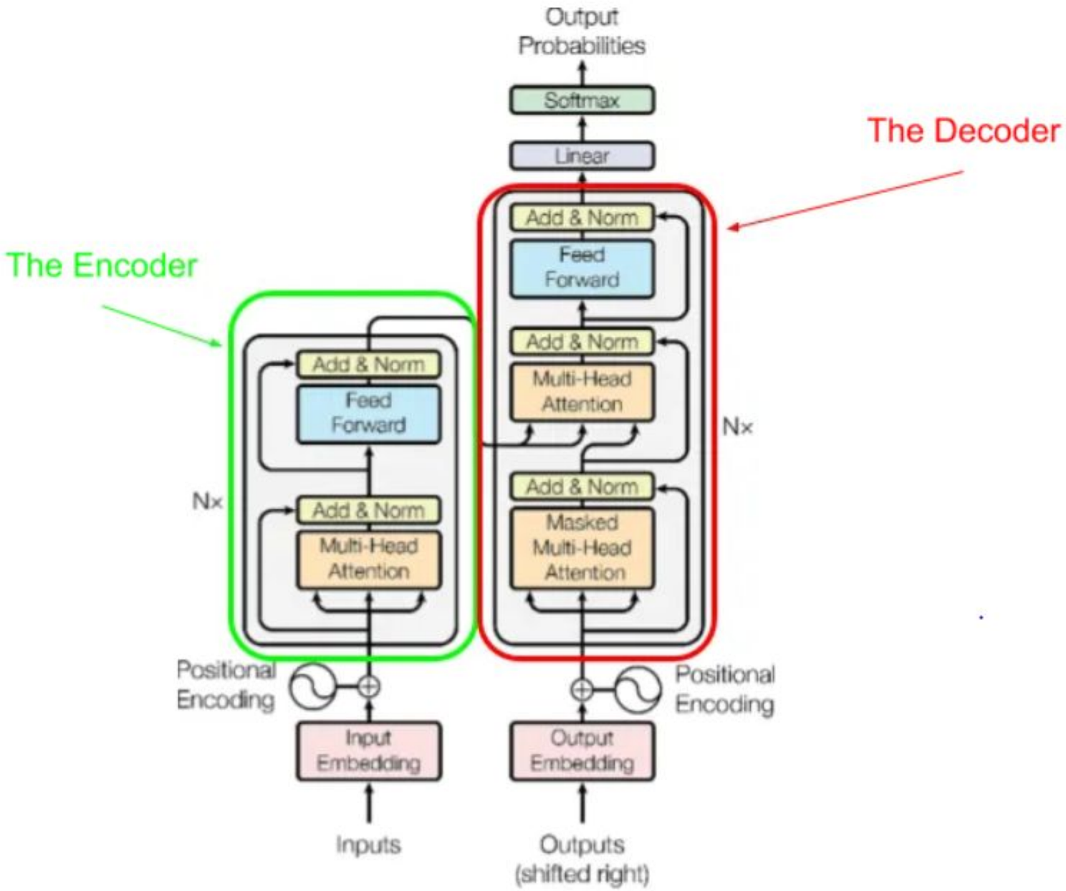
Class Meeting 16: Project Presentations and Conclusion (1 session)
Students present their NLP projects
Recap of key takeaways from the course
Discuss further resources for NLP enthusiasts
Course conclusion and feedback

Agenda

- Introduction to Large Language Model

LLM

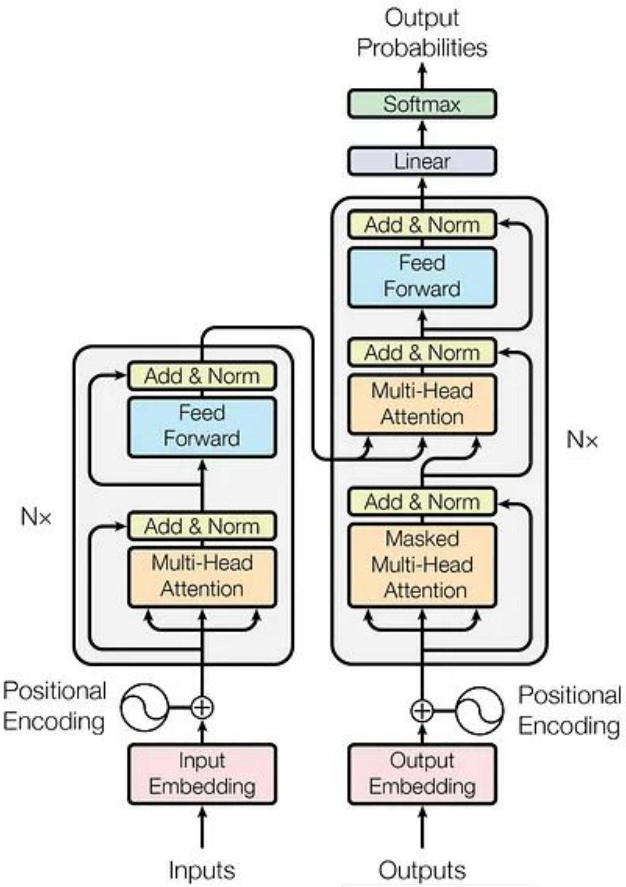
Transformer Architecture



Transformer Architecture

BERT

Encoder



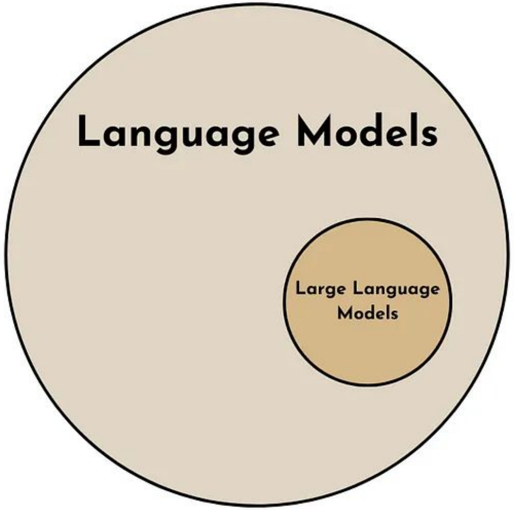
GPT

Decoder

What makes llm “Large”

When I heard the term “Large Language Model,” one of my first questions was, how is this different from a “regular” language model?

A language model is more generic than a large language model. Just like all squares are rectangles but not all rectangles are squares. **All LLMs are language models, but not all language models are LLMs.**



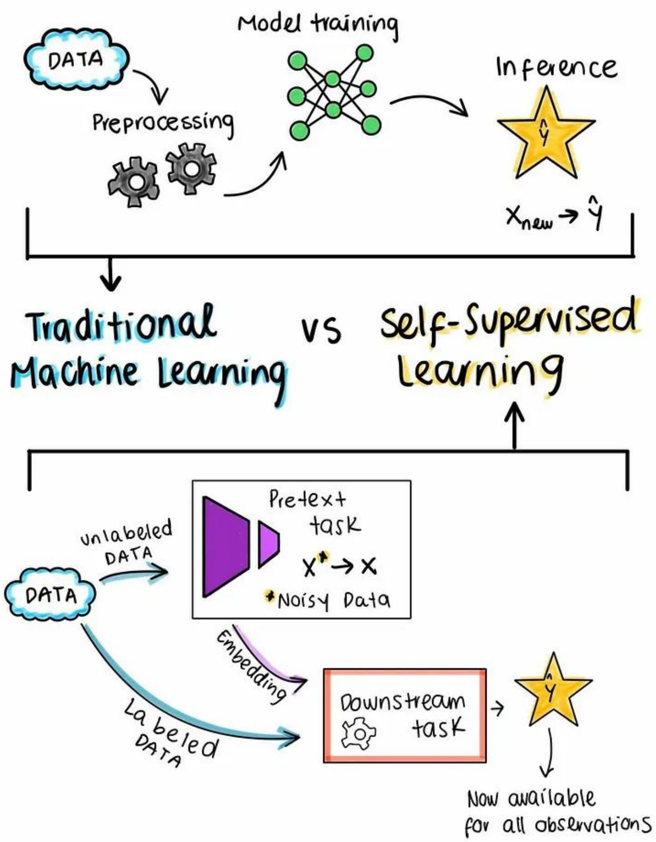
Types of LLM

There are 2 key properties that distinguish LLMs from other language models. One is quantitative, and the other is qualitative.

- Quantitatively, what distinguishes an L**LM is the number of parameters used in the model**. Current LLMs have on the order of 10–100 billion parameters .

- Qualitatively, something **remarkable happens when a language model becomes "large."** It exhibits so-called emergent properties e.g. zero-shot learning . These are properties that seem to suddenly appear when a language model reaches a sufficiently large size.

Self Supervised Learning



Zero Shot Learning

The major innovation of GPT-3 (and other LLMs) is that it is capable of zero-shot learning in a wide variety of contexts. **This means ChatGPT can perform a task even if it has not been explicitly trained to do it.**

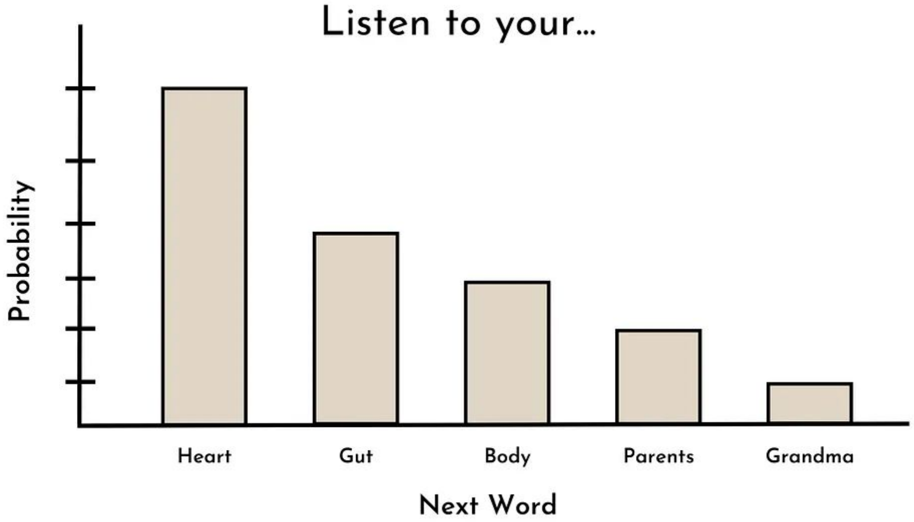
For instance, if you wanted a computer to do language translation, sentiment analysis, and identify grammatical errors. Each of these tasks would require a specialized model trained on a large set of labeled examples. Now, however, **LLMs can do all these things without explicit training.**

```
##put your from skllm import DynamicFewShotGPTClassifier

X = [
    "I love reading science fiction novels, they transport me to other worlds.",
    "A good mystery novel keeps me guessing until the very end.",
    "Historical novels give me a sense of different times and places.",
    "I love watching science fiction movies, they transport me to other galaxies.",
    "A good mystery movie keeps me on the edge of my seat.",
    "Historical movies offer a glimpse into the past.",
]

y = ["books", "books", "books", "movies", "movies", "movies"]

query = "I have fallen deeply in love with this sci-fi book; its unique blend of science and fiction has me spellbound."

clf = DynamicFewShotGPTClassifier(n_examples=1).fit(X, y)

prompt = clf._get_prompt(query)
print(prompt)
```

How do LLMs work?

The core task used to train most state-of-the-art LLMs is **word prediction**. In other words, given a sequence of words, **what is the probability distribution of the next word**?

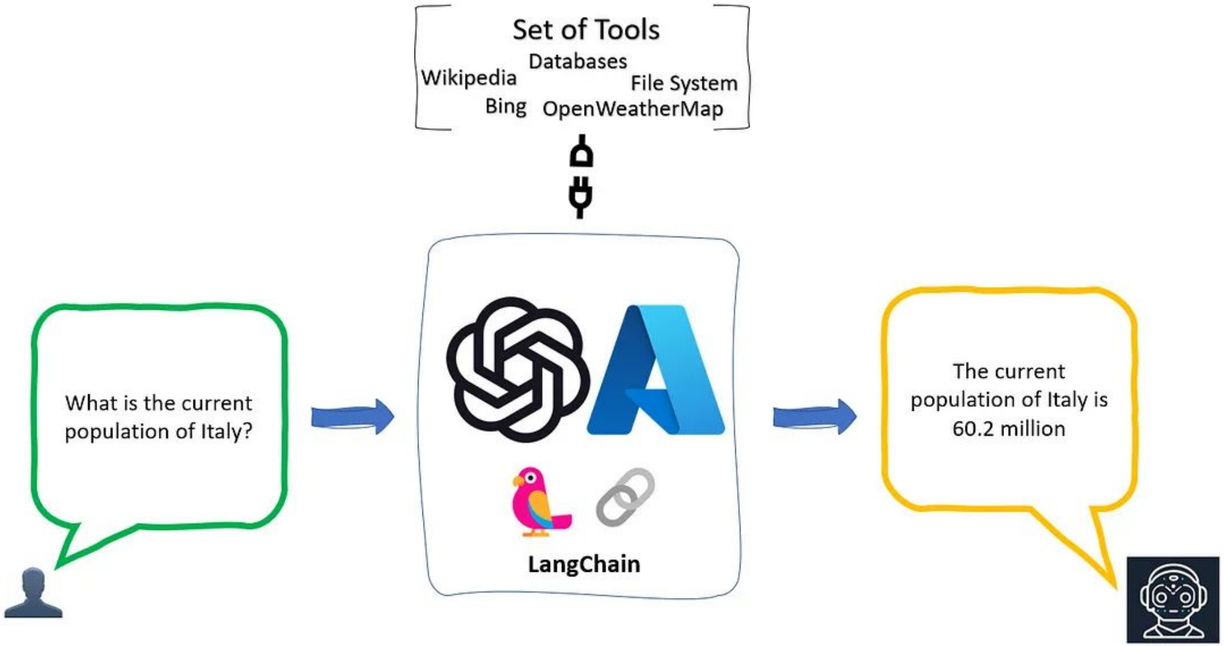
For example, given the sequence “Listen to your ____,” the most likely next words might be: heart, gut, body, parents, grandma, etc. This might look like the probability distribution shown below.



Langchain

Agent

Agents can be seen as applications powered by LLMs and integrated with a set of tools **like search engines, databases, websites, and so on**. Within an agent, the LLM **is the reasoning engine that, based on the user input, is able to plan and execute a set of actions that are needed to fulfill the request.**



Langchain Example

```
agent.run("which are the Top 10 countries with the highest total_ghg in year=2018 excluding NaN. Show the total_ghg values")
```

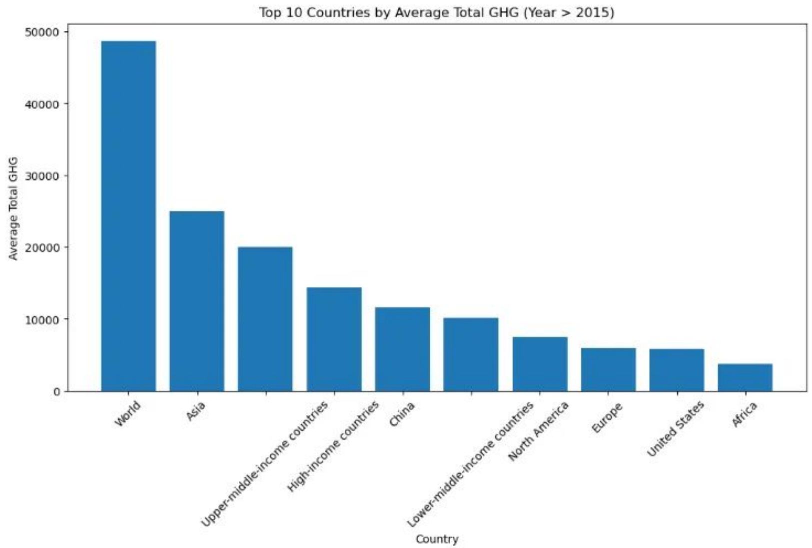
```
> Entering new  chain...

Invoking: `python_repl_ast` with `df[df['year'] == 2018].sort_values('total_ghg', ascending=False)[['country', 'total_ghg']].he
ad(10)`


                           country  total_ghg
50078                        World  49368.039
3050                          Asia  25456.090
48599  Upper-middle-income countries  20273.920
20312         High-income countries  14425.740
9422                         China  11821.660
26839  Lower-middle-income countries  10372.980
33433                North America   7630.500
14103                       Europe   5980.720
48155                United States   5892.370
440                         Africa   3688.830The top 10 countries with the highest total greenhouse gas (GHG) emissions in th
e year 2018, excluding NaN values, are:

1. World - 49368.039
2. Asia - 25456.090
3. Upper-middle-income countries - 20273.920
4. High-income countries - 14425.740
5. China - 11821.660
6. Lower-middle-income countries - 10372.980
7. North America - 7630.500
8. Europe - 5980.720
9. United States - 5892.370
10. Africa - 3688.830

> Finished chain.
```

```
'The top 10 countries with the highest total greenhouse gas (GHG) emissions in the year 2018, excluding NaN values, are:\n\n1.
World - 49368.039\n2. Asia - 25456.090\n3. Upper-middle-income countries - 20273.920\n4. High-income countries - 14425.740\n5.
China - 11821.660\n6. Lower-middle-income countries - 10372.980\n7. North America - 7630.500\n8. Europe - 5980.720\n9. United S
tates - 5892.370\n10. Africa - 3688.830'
```

Langchain Example

```
In [44]: agent.run("show a bar chart by Top 10 countr of average total_ghg where year > 2015 sorted in descending order")
```

```
> Entering new  chain...
```




```
Invoking: `python_repl_ast` with `import matplotlib.pyplot as plt

# Filter the dataframe for year > 2015
filtered_df = df[df['year'] > 2015]

# Group by country and calculate the average total_ghg
grouped_df = filtered_df.groupby('country')['total_ghg'].mean()

# Sort the values in descending order and select the top 10
top_10_countries = grouped_df.sort_values(ascending=False).head(10)

# Create the bar chart
plt.figure(figsize=(12, 6))
plt.bar(top_10_countries.index, top_10_countries.values)
plt.xlabel('Country')
plt.ylabel('Average Total GHG')
plt.title('Top 10 Countries by Average Total GHG (Year > 2015)')
plt.xticks(rotation=45)
plt.show()`
```

Retrieval-Augmented Generation (RAG)

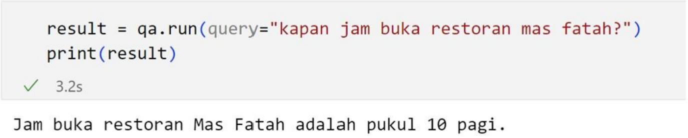
Retrieval-Augmented Generation (RAG) is the concept to **provide LLMs with additional information from an external knowledge source**. This allows them to generate more accurate and **contextual answers while reducing hallucinations**.

Retrieval-Augmented Generation (RAG)

State-of-the-art LLMs are trained on large amounts of data to achieve a broad spectrum of general knowledge stored in the neural network's weights (parametric memory). However, prompting an LLM to generate a completion that requires knowledge that was not included in its training data, such as newer, proprietary, or domain-specific information, **can lead to factual inaccuracies (hallucinations)**, as illustrated in the following screenshot:

Default GPT

RAG

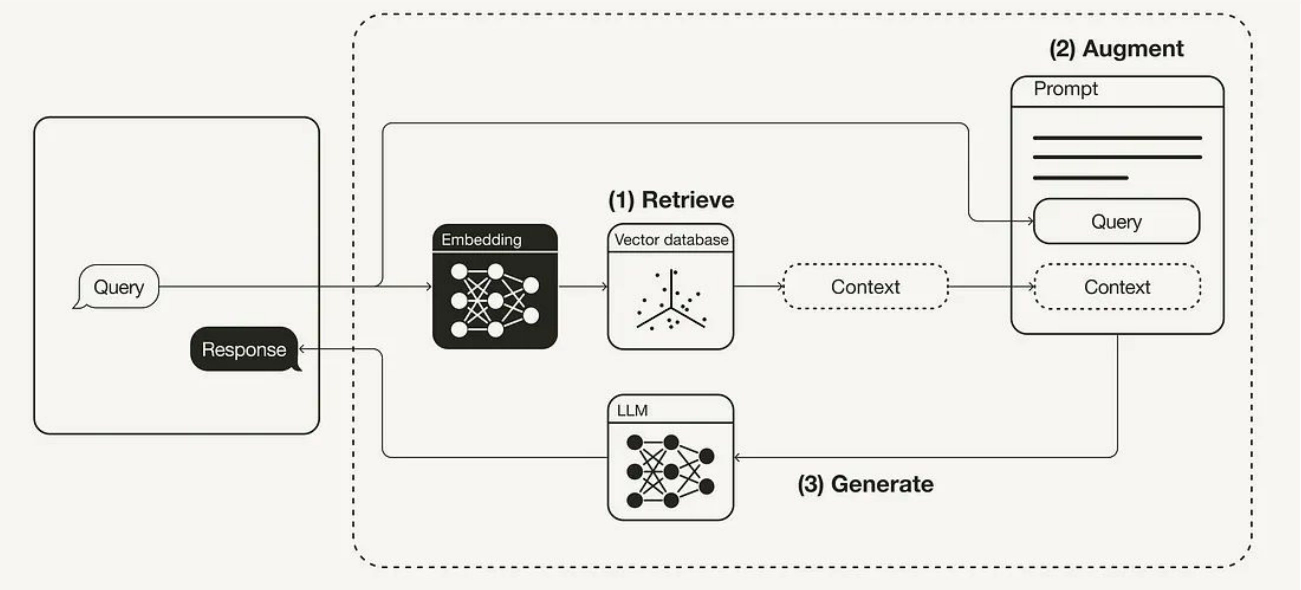


Retrieval-Augmented Generation (RAG)

Traditionally, neural networks are adapted to domain-specific or proprietary information by fine-tuning the model. **Although this technique is effective, it is also compute-intensive, expensive, and requires technical expertise, making it less agile to adapt to evolving information**.

In simple terms, RAG is to LLMs what an open-book exam is to humans. In an open-book exam, students are allowed to bring reference materials, such as textbooks or notes, which they can use to look up relevant information to answer a question. The idea behind an open-book exam is that the test focuses on the students’ reasoning skills rather than their ability to memorize specific information.

Retrieval-Augmented Generation (RAG)



1. **Retrieve:** The user query is used to retrieve relevant context from an external knowledge source. For this, the user query is embedded with an embedding model into the same vector space as the additional context in the vector database. This allows to perform a similarity search, and the top k closest data objects from the vector database are returned.
2. **Augment:** The user query and the retrieved additional context are stuffed into a prompt template.
3. **Generate:** Finally, the retrieval-augmented prompt is fed to the LLM.

Python Time

https://github.com/muhajirakbarhsb/NLP_class_2023/blob/main/Week_Class_11.ipynb