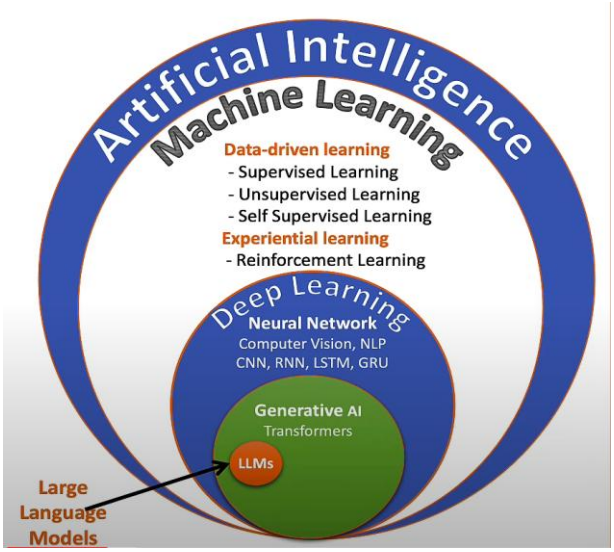
Ai is a giant thing that holds a lot of fields that we hear in this era



We talk for example of Machine learning which allow computers to learn and make predictions or decisions based on data or experience.

- ➔ Data driven learning :

  . **Supervised learning** in which we give the computer a huge amount of data labeled say

  Example: Diabetic patients dataset where each patient's data is labeled as either positive (diabetic) or negative (non-diabetic).

  . **Unsupervised learning** involves training the model on data without labeled responses. The algorithm looks for hidden patterns or structures in the data, aiming to learn more about the data's underlying structure.

  Example: Clustering similar patient profiles based on health metrics without predefined labels.

  . Self supervised learning where the labels are generated automatically from the input data itself. It involves training a model to predict a part of the input from the rest of it.

  Example: Predicting the missing word in a sentence (masked language modeling) or predicting the next frame in a video sequence.

- ➔ Experience driven learning :

  . **Reinforcement learning :** agent learns to make decisions by interacting with an environment. The agent receives feedback in the form of rewards or penalties for its actions, which helps it learn optimal behaviors over time.

  Example: Teaching a robot to navigate a maze where it learns the best path through trial-and-error, adjusting its actions based on rewards received (e.g., reaching the goal) or penalties (e.g., hitting obstacles).

And also when we talk about deep learning it is a subset of machine learning that manage to hold an important amount of data as input using some algorithms like neural networks which are inspired by the human brain's structure, consisting of interconnected nodes (neurons) organized in layers.

Then we move to Genrative AI which is our main thing in this research:

Generative AI

Generative AI refers to a type of Artificial Intelligence that is capable of generating new data, such as images, music, or text, based on a set of training data

Reasons that have led to a significant acceleration in the pace of research and development in generative AI.

- Powerful Language Models :
  - For NLP : (GPT, BERT) :
    Machine translation, Text summarization, Sentiment analysis, Conversational AI
  - For Computer Vision : GANs, Stable Diffusion, DALL-E, Midjourney
- Abundant Data:
  - In 2022, users sent around 650 million Tweets per day.
  - In 2022, 333.2 billion emails were sent every day.
- Hardware Advancements :

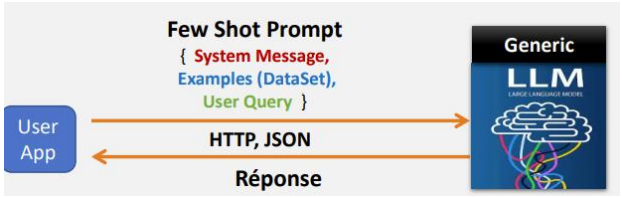
  GPU & TPU , Cloud Computing

The thing is generative Ai uses models that we call large language models that are models trained on vast amounts of text data to understand and generate human-like text. These models functions with a famous algorithm named transformers that is published in 2017 by some google employees in an article called **Attention is all you need** and we can see it's impact on well-known models like BERT or GPT :

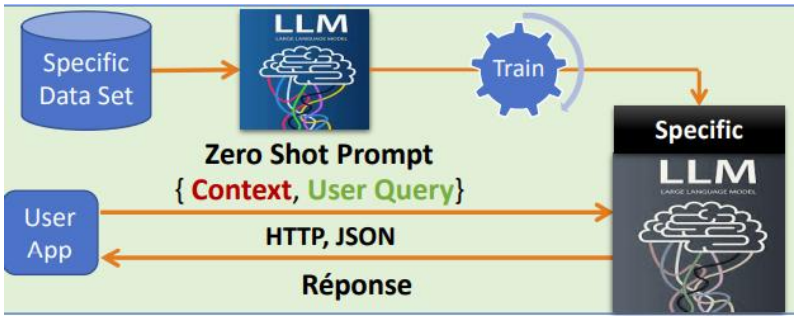
- **BERT (Bidirectional Encoder Representations from Transformers)**:
  - BERT, developed by Google in 2018, is a bidirectional Transformer-based model. It captures context from both directions in a sequence, significantly improving the understanding of language semantics. BERT has been widely used for tasks such as text classification, named entity recognition, and question answering.
- **GPT (Generative Pre-trained Transformer)**:
  - GPT, developed by OpenAI, is another variant of the Transformer model. It employs a decoder-only architecture, making it suitable for generating human-like text based on learned patterns from vast amounts of text data. GPT models have been used for tasks like text generation, dialogue systems, and language translation.

The interesting part comes when we wan't to use these pretrained models on our specific use case what we must known is there are 3 ways to do it:

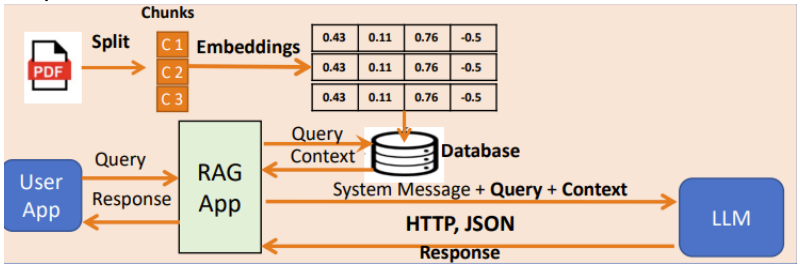
1- Few Shot learning:
where a model is trained to generalize from a few examples (shots) of each class or task. It enables the model to learn from a small amount of labeled data, which is crucial when specific use cases have limited annotated datasets.



2- Fine tuning involves taking a pretrained model (like BERT or GPT) that has been trained on a large dataset and further training it on a smaller, domain-specific dataset.



3- RAG(Retrieval augmented generation) combines generative AI with retrieval-based methods. It uses a retrieval mechanism to augment the generation process, allowing the model to retrieve relevant information before generating a response.



Example of use : trying to use the RAG method using a pdf file :

$ In this example we use pinecone to store embeddings of text chunks

```
from langchain import PromptTemplate
from langchain.chains import RetrievalQA
from langchain.embeddings import HuggingFaceEmbeddings
from langchain.vectorstores import Pinecone
import pinecone
from langchain.document_loaders import PyPDFLoader, DirectoryLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.prompts import PromptTemplate
from langchain.llms import CTransformers
```

```
PINECONE_API_KEY = "038f965a-07af-486e-ab43-d1b21f18a91d"
PINECONE_API_ENV = "gcp-starter"
import os
os.environ['PINECONE_API_KEY'] = '038f965a-07af-486e-ab43-d1b21f18a91d'
```

$ this function extracts data from the pdf given in parameter

```
#Extract data from the PDF
def load_pdf(data):
    loader = DirectoryLoader(data,
                             glob="*.pdf",
                             loader_cls=PyPDFLoader)

    documents = loader.load()

    return documents
```

$ This function create text chunks

```
#Create text chunks
def text_split(extracted_data):
    text_splitter = RecursiveCharacterTextSplitter(chunk_size = 500, chunk_overlap = 20)
    text_chunks = text_splitter.split_documents(extracted_data)

    return text_chunks
```

$ This function downloads a hugging face model to do the embeddings of text chunks

```
#download embedding model
def download_hugging_face_embeddings():
    embeddings = HuggingFaceEmbeddings(model_name="sentence-transformers/all-MiniLM-L6-v2")
    return embeddings
```

$ Create instance of Pinecone class

```
from langchain_pinecone import PineconeVectorStore

#Initializing the Pinecone
pc = pinecone.Pinecone(api_key=PINECONE_API_KEY,
                       environment=PINECONE_API_ENV)
```

$ Creating embeddings (numerical vectors so that we can store then in the pinecone semantic index)

```
index_name="chatbot"
vectorstore_from_texts = PineconeVectorStore.from_texts(
        [t.page_content for t in text_chunks],
        index_name=index_name,
        embedding=embeddings
    )
```

$ Creating the prompt template

```
prompt_template="""
Use the following pieces of information to answer the user's question.
If you don't know the answer, just say that you don't know, don't try to make up an answer.

Context: {context}
Question: {question}

Only return the helpful answer below and nothing else.
Helpful answer:
"""
```

```
PROMPT=PromptTemplate(template=prompt_template, input_variables=["context", "question"])
chain_type_kwargs={"prompt": PROMPT}
```

$ initialize a language model using the `CTransformers library (in this case llama)`

```
llm=CTransformers(model="model/llama-2-7b-chat.ggmlv3.q4_0.bin",
                  model_type="llama",
                  config={'max_new_tokens':512,
                          'temperature':0.8})
```

$ it retrieves answers from the knowledge base and it gives the best answer among the 2

```
qa=RetrievalQA.from_chain_type(
    llm=llm,
    chain_type="stuff",
    retriever=vectorstore_from_texts.as_retriever(search_kwargs={'k': 2}),
    return_source_documents=True,
    chain_type_kwargs=chain_type_kwargs)
```

$ Example of use

```
while True:
    user_input=input(f"Input Prompt:")
    result=qa({"query": user_input})
    print("Response : ", result["result"])
```

```
Input Prompt: what is acne
```

```
C:\Users\hamza\AppData\Roaming\Python\Python312\site-packages\langchain_core\_api\deprecation.py:139: LangChainDeprecationWarning: The method `
ll__` was deprecated in langchain 0.1.0 and will be removed in 0.3.0. Use invoke instead.
  warn_deprecated(
```

```
Response :  Acne is a common skin disease characterized by pimples on the face, chest, and back. It occurs when the pores of the skin become cl
oil, dead skin cells, and bacteria.
```