

8630 Generative AI Development and Engineering



Web Age Solutions
USA: 1-877-517-6540
Canada: 1-877-812-8887
Web: <http://www.webagesolutions.com>

The following terms are trademarks of other companies:

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

IBM, WebSphere, DB2 and Tivoli are trademarks of the International Business Machines Corporation in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

For customizations of this book or other sales inquiries, please contact us at:

USA: 1-877-517-6540, email: getinfousa@webagesolutions.com

Canada: 1-877-812-8887 toll free, email: getinfo@webagesolutions.com

Copyright © 2023 Web Age Solutions

This publication is protected by the copyright laws of Canada, United States and any other country where this book is sold. Unauthorized use of this material, including but not limited to, reproduction of the whole or part of the content, re-sale or transmission through fax, photocopy or e-mail is prohibited. To obtain authorization for any such activities, please write to:

Web Age Solutions
220 Yonge Street, Suite 218B
Toronto, Ontario. M5B 2H1

TOC

- Ch. 1: Introduction to Generative AI ...4
- Ch. 2: Generative AI Architecture ...40
- Ch. 3: Tuning Generative AI Models ...81
- Ch. 4: Case Studies and Real-World Applications [only Text Generation] ...135
- Ch. 5: Security ...201
- Ch. 6: Evaluation and Optimization of Generative AI Models ...237
- Ch. 7: Bedrock/Foundational Models ...264
- Ch. 8: Tuning GenAI Models [Transfer Learning section, from end] ...291
- Ch. 9: ChatBots ...326
- Ch. 10: Building Generative AI Applications (part 1) ...349

Introduction to Generative AI

Chapter 1

Agenda

- Generative AI's Roots in Machine learning
- Understanding Generative models
- Contrasting Generative and Discriminative Models
- The original LLM models – from BERT to GPT
- Current Cloud- and Offline-Based LLM's
- Conclusion

Generative AI's Roots in Machine Learning

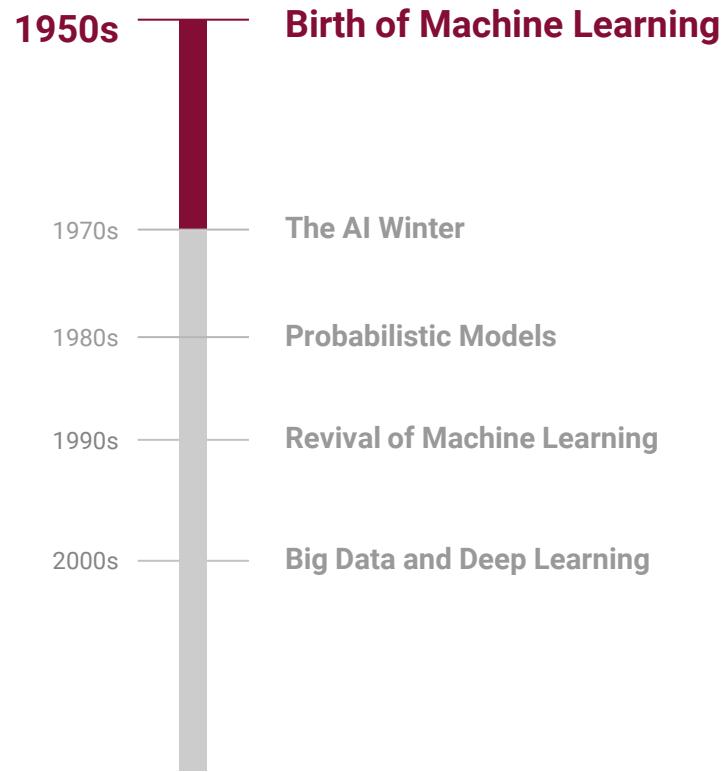
- **What is ML?**
 - **Machine learning** (ML) is a subset of Artificial Intelligence (AI) that allows computers to **learn from data without any programming intervention**.
 - ML algorithms **use data to identify patterns** and learn from them, making it possible to automate tasks and predict outcomes.

How Machine Learning Led to Generative AI

1950s - 1960s

Birth of ML

- 1959 - Arthur Samuel coins the term "Machine Learning". His checkers playing program was an early examples of a self-learning program.
- 1959 - Frank Rosenblatt creates the **Perceptron** - the building block of neural networks today

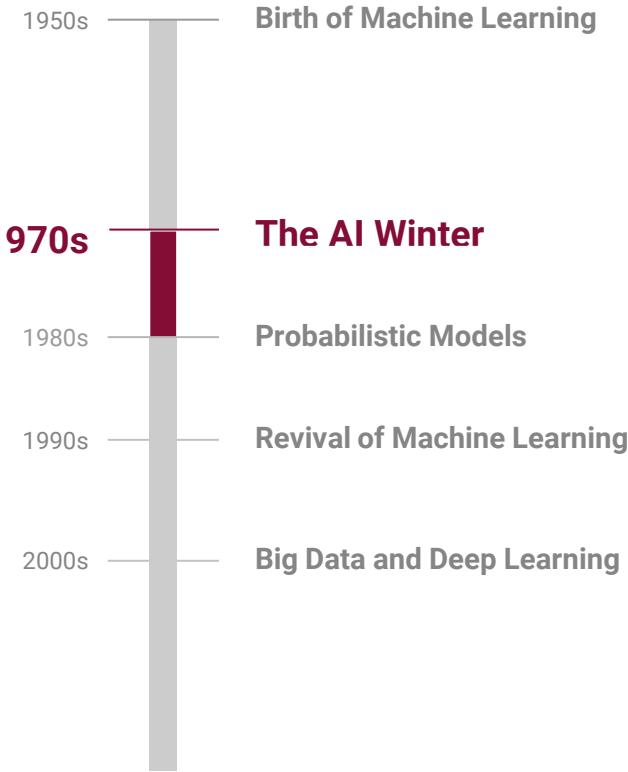


How Machine Learning Led to Generative AI

1970s - 1980s

The AI Winter

- The AI Winter and the **Emergence of Rule-Based Systems**
- **Reduced funding for ML** research, focus on rule-based systems

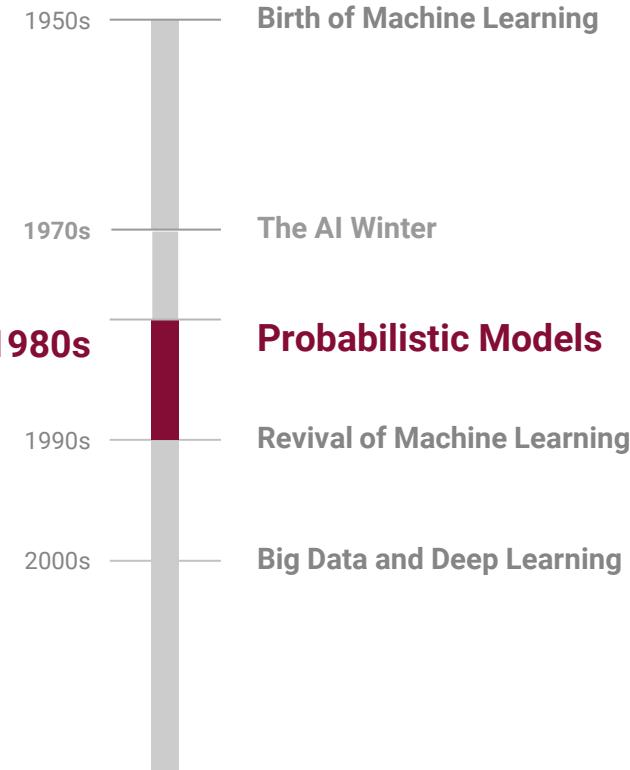


How Machine Learning Led to Generative AI

1980s - 1990s

Probabilistic Models

- researchers started exploring **probabilistic models** for generative tasks.
- Hidden Markov Models (HMMs) and Bayesian networks were used for **speech recognition, language modeling, and natural language processing.**

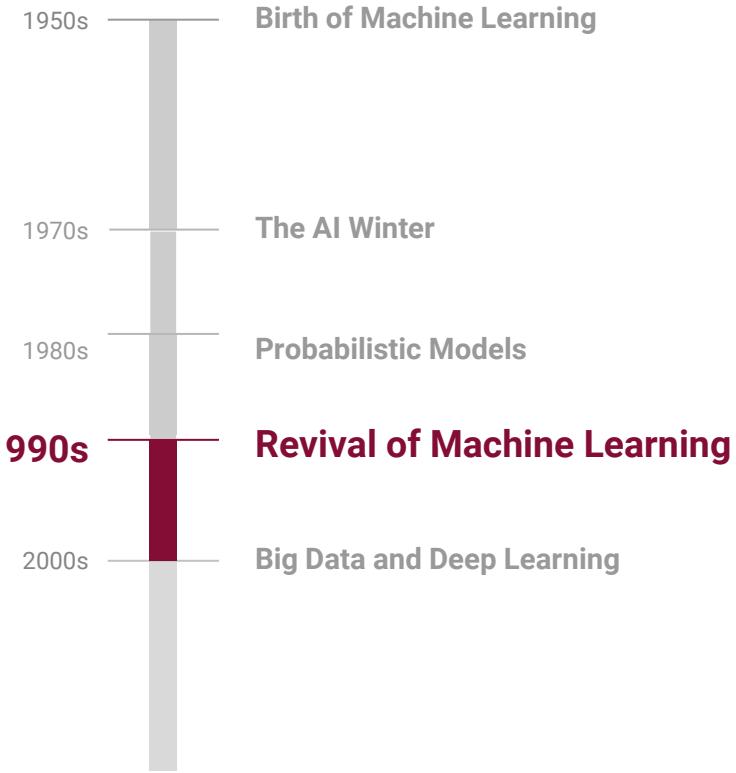


How Machine Learning Led to Generative AI

1990s - 2000s

Revival of ML

- **ML Algorithms** (like Decision Trees, K-Nearest Neighbors, and Support Vector Machines) **revive interest/funding in ML**
- Ensemble methods like Random Forest and Boosting.

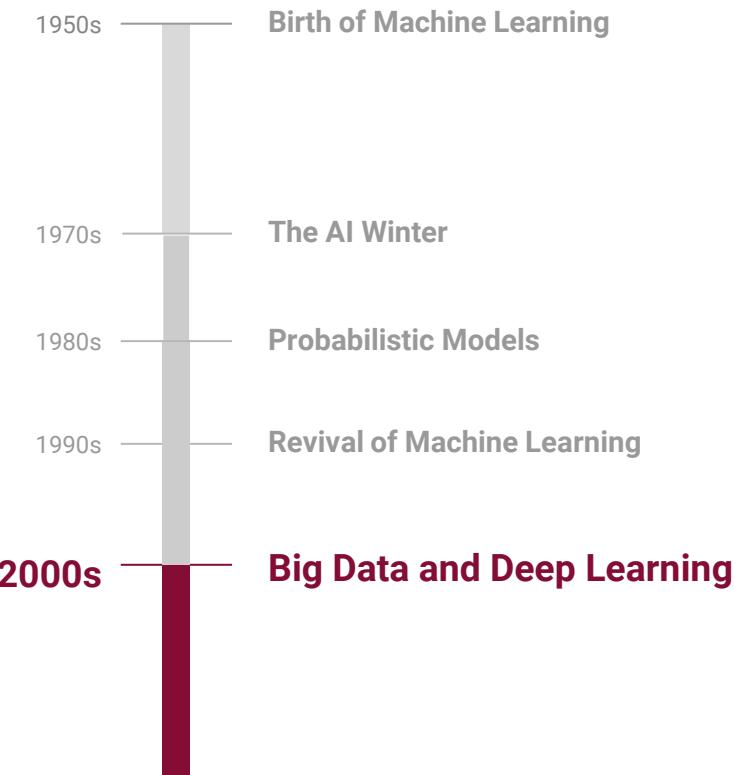


How Machine Learning Led to Generative AI

2000s - 2010s

Big Data and Deep Learning

- With the Internet and Big Data, ML becomes crucial to generate insight
- **Neural Network advances:** Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN)
- **Deep Learning develops**

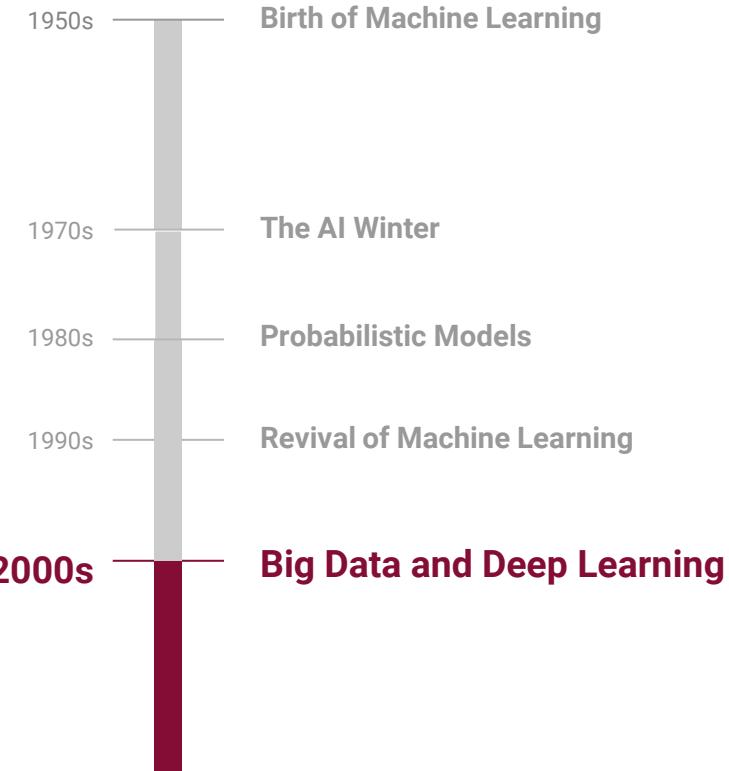


How Machine Learning Led to Generative AI

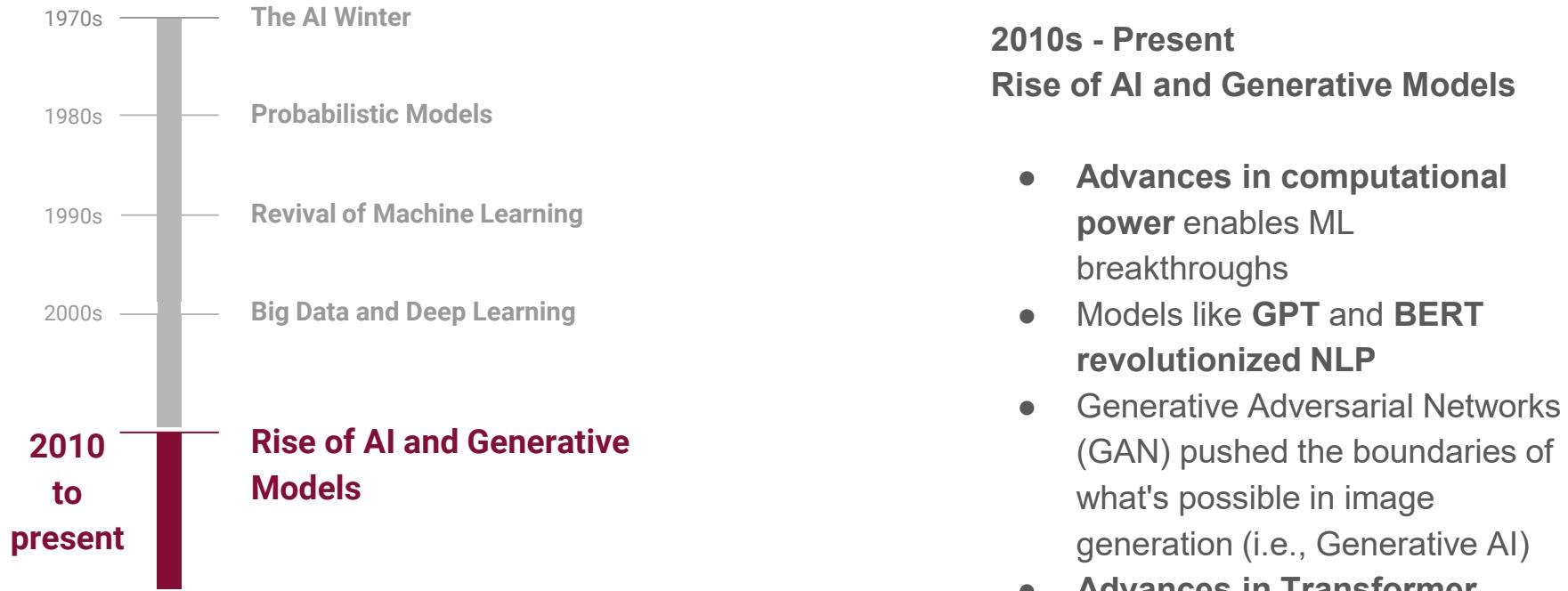
2000s - 2010s

Big Data and Deep Learning

- **Variational Autoencoders (VAEs) were introduced,** which marked a significant advancement in generative AI.
- **VAEs provided a probabilistic framework for generating new data samples,** enabling various applications in image generation and data compression



How Machine Learning Led to Generative AI



Understanding Generative models

- **What are Generative Models?**
 - **Definition**
 - **Types**
 - **Examples**
 - **Applications**
- **Generative models** are a class of machine learning models that aim to **generate new data samples** that resemble the distribution of a given training dataset.
- These models learn the underlying patterns and structures present in the training data and then use that knowledge to **produce new, synthetic data samples** that are similar to the real data (as opposed to classifying or predicting)

Understanding Generative models

- **What are Generative Models?**
 - Definition
 - **Types**
 - Examples
 - Applications
- **Transformer Models:** Attention based models, common in LLM's, understand context very well. Also used in Image GPT.
- **Generative Adversarial Networks (GANs):** Consists of two models. A Generator creates new data instances, while a Discriminator evaluates them for authenticity/quality

Understanding Generative models

- **What are Generative Models?**
 - Definition
 - **Types**
 - Examples
 - Applications
- **Variational Autoencoders (VAEs)** - models that use encoder and decoder neural networks to learn a compact representation of input data in a lower-dimensional latent space, enabling the generation of new data samples with similar characteristics to the training data.
- **Normalizing Flow Models** - models that use invertible transformations to map a simple distribution (e.g., Gaussian) to a complex data distribution, enabling efficient sampling and exact likelihood computation.

Understanding Generative models

- **What are Generative Models?**
 - Definition
 - **Types**
 - Examples
 - Applications
- **Energy-based models (EBMs)** - model that assigns an energy score to each input sample, and the probability of the sample is inversely proportional to its energy, allowing the model to capture complex dependencies and patterns in the data.
- **AutoRegressive models (AR):** These models generate sequences by modeling the probability of each subsequent item based on the items that preceded it.
- **...and others**

Understanding Generative models

- **What are Generative Models?**
 - Definition
 - Types
 - **Examples**
 - Applications
- **DCGAN (Deep Convolutional Generative Adversarial Network)** - a variant of Generative Adversarial Networks that uses deep convolutional neural networks to generate high-resolution and realistic images.
- **PixelRNN and PixelCNN** - autoregressive models capable of generating images pixel by pixel, where each pixel's color depends on previously generated pixels.

Understanding Generative models

- **What are Generative Models?**
 - Definition
 - Types
 - **Examples**
 - Applications
- **GPT** - a language model that uses a transformer architecture to generate coherent and contextually appropriate text, often used for tasks like text generation and completion.
- **DRAW (Deep Recurrent Attentive Writer)**:
uses recurrent neural networks and attention mechanisms to generate images in a step-by-step manner.

Understanding Generative models

- **What are Generative Models?**
 - Definition
 - Types
 - **Examples**
 - Applications
- **Flow++** - flow-based model used for density estimation and image generation, capable of producing high-quality samples.
- **BERT**

Understanding Generative models

- **What are Generative Models?**
 - Definition
 - Types
 - Examples
 - Applications
- **Image Generation and Synthesis -** create realistic images of objects, scenes, and even people. It has applications in art, design, and entertainment industries for generating visual content.



Understanding Generative models

- **What are Generative Models?**
 - Definition
 - Types
 - Examples
 - Applications
- **Text Generation and Language Modeling** - language models and transformers can generate human-like text, which has applications in natural language processing, chatbots, and content creation.



Understanding Generative models

- **What are Generative Models?**
 - Definition
 - Types
 - Examples
 - **Applications**
- **Code Generation and Language Modeling** - language models are also able to generate code from text prompts or other code. Language models can be trained on programming languages too, not just human languages.



Understanding Generative models

- **What are Generative Models?**
 - Definition
 - Types
 - Examples
 - Applications
- **Data Augmentation** - augment training datasets by generating new data samples, helping improve the performance and generalization of machine learning models.
- **Text Generation and Language Modeling** - pharmaceutical research to design and discover new molecules and drugs, accelerating drug development processes.
- **... and others**

Difference between Generative and Discriminative Models

- **Differences**
 - **Objective**
 - Data Utilization
 - Complexity
 - Applications
 - Handling Imbalanced Data
- **Discriminative Models** - focus on modeling the conditional probability distribution of the labels given the input data. They are **primarily** used for making **predictions** and classifying new data samples.
- **Generative Models** - aim to model the joint probability distribution of the input data and labels. They learn the underlying data distribution and can **generate new data** samples.

Difference between Generative and Discriminative Models

- **Differences**
 - Objective
 - **Data Utilization**
 - Complexity
 - Applications
 - Handling Imbalanced Data
- **Discriminative Models** - use only the input data and corresponding labels **during training**. They focus on learning the decision boundary between different classes to classify new data samples accurately.
- **Generative Models** - use both the input data and labels during training. They aim to **capture the complete data distribution**, which allows them to generate new data samples.

Difference between Generative and Discriminative Models

- **Differences**
 - Objective
 - Data Utilization
 - **Complexity**
 - Applications
 - Handling Imbalanced Data
- **Discriminative Models** - generally **simpler** than generative models because they focus on modeling the decision boundary between classes rather than the complete data distribution.
- **Generative Models** - tend to be **more complex** than discriminative models because they need to model the entire data distribution. This complexity can make them **computationally expensive** and **require more data for training**.

Difference between Generative and Discriminative Models

- **Differences**
 - Objective
 - Data Utilization
 - Complexity
 - **Applications**
 - Handling Imbalanced Data
- **Discriminative Models** - commonly used for tasks that involve **classification and prediction**, such as image recognition, sentiment analysis, and natural language processing.
- **Generative Models** - often used for tasks that involve **generating new data samples**, such as image synthesis, text generation, and music composition. They can also be used in unsupervised learning and semi-supervised learning scenarios.

Difference between Generative and Discriminative Models

- **Differences**
 - Objective
 - Data Utilization
 - Complexity
 - Applications
 - **Handling Imbalanced Data**
- **Discriminative Models** - may struggle with **imbalanced** data since they focus on learning the decision boundary, and the model's performance may be biased towards the majority class.
- **Generative Models** - can handle imbalanced data well since they model the entire data distribution. They are **less sensitive** to imbalanced class distributions during training.

The original LLM models – from BERT to GPT

- **LLM models**
 - **BERT**
 - **GPT-2**
 - **GPT-3**
 - **GPT-3.5 and beyond**
- Released in 2018 by **Google AI**.
- Introduced **bidirectional context** to pre-training language models.
- Utilized a masked language model (**MLM**) pre-training objective.
- **Pre-trained** on massive amounts of **text data**.
- **State-of-the-art performance** on various NLP tasks.

The original LLM models – from BERT to GPT

- **LLM models**
 - BERT
 - **GPT-2**
 - GPT-3
 - GPT-3.5 and beyond
- Released in **2019** by **OpenAI**.
- **Left-to-right unidirectional** training objective.
- **1.5 billion parameters**.
- **Autoregressive decoding** during generation.
- Impressive text generation capabilities.

The original LLM models – from BERT to GPT

- **LLM models**
 - BERT
 - GPT-2
 - **GPT-3**
 - GPT-3.5 and beyond
- Released in **2020** by **OpenAI**.
- Scaled up the model size to **175 billion parameters**.
- Largest publicly known LLM at the time.
- **Few-shot and zero-shot learning capabilities.**
- Performance across a wide range of tasks.

The original LLM models – from BERT to GPT

- **LLM models**
 - BERT
 - GPT-2
 - GPT-3
 - **GPT-3.5 and beyond**
- Continuation of the GPT-3 series.
- Improvements in **language understanding** and generation capabilities.
- Latest release in the is **GPT-4**, launched in 2023.
- Expected that GPT-4 will continue the pattern of **increasing model size and performance**.

Exponential growth in parameter number

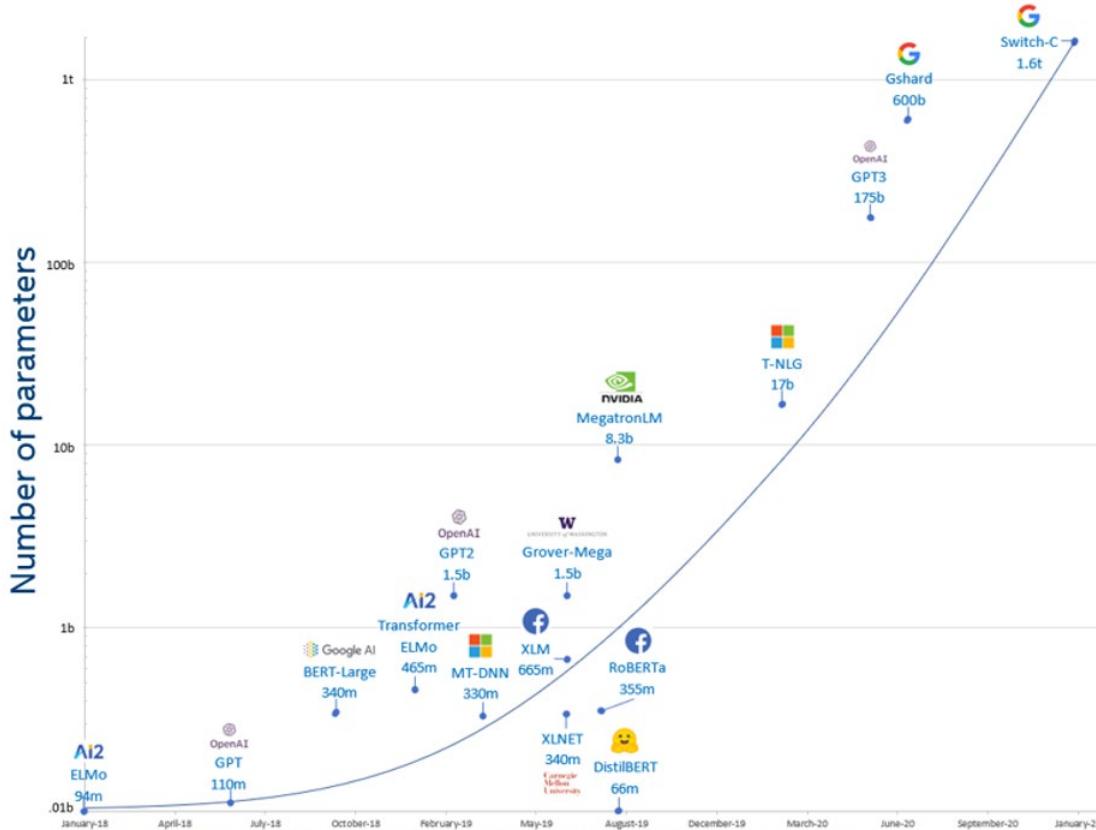
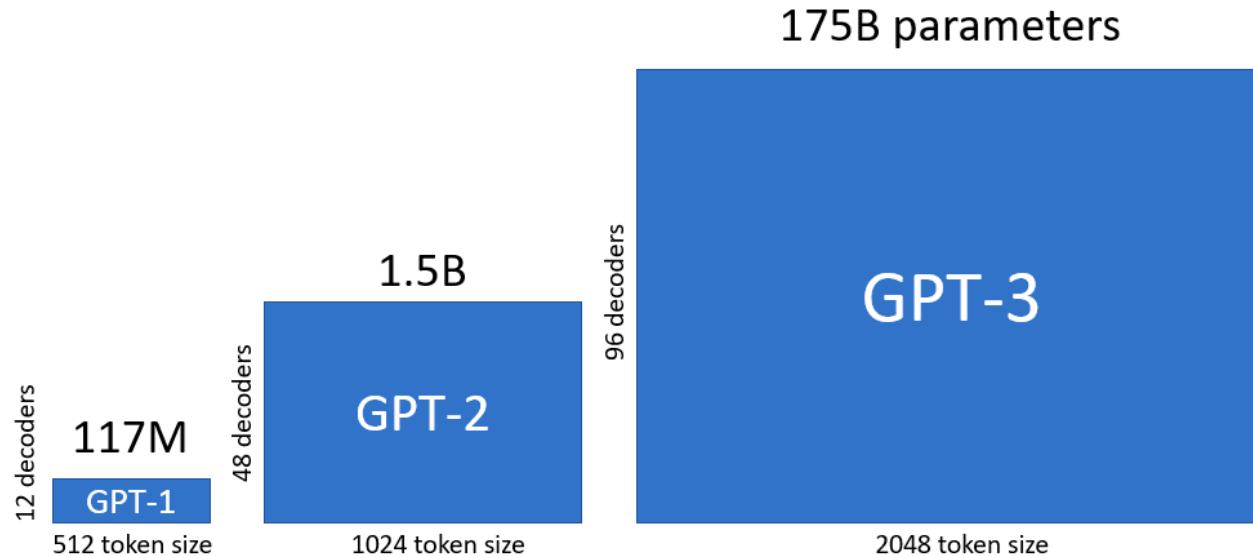
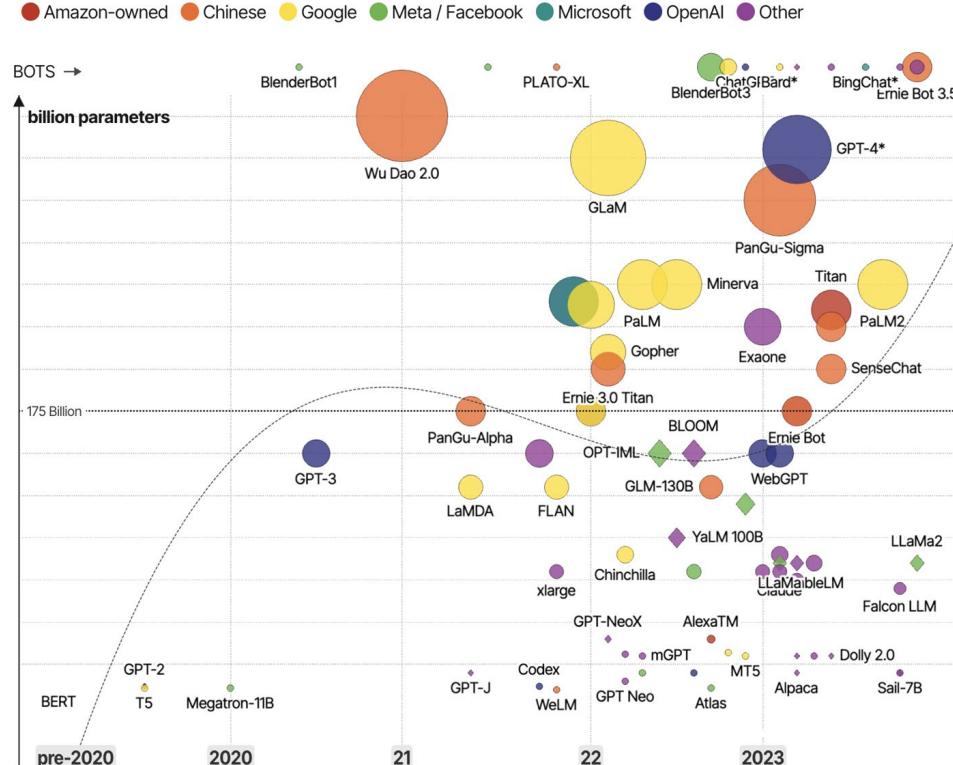


Figure 1: Exponential growth of number of parameters in DL models

Model complexity and exponential scaling



New players emerging with very large models



David McCandless, Tom Evans, Paul Barton
Information is Beautiful // UPDATED 27th Jul 23

36

source: news reports, [LifeArchitect.ai](#)
* = parameters undisclosed // see [the data](#)

Cloud-Based LLM's

Some of the cloud-based LLM's available include:

- OpenAI's GPT3 and GPT4
- Google's Palm 2
- Anthropic's Claude v2
- Cohere's Cohere LLM (same name as the company)

Offline-Based LLM's

Some of the offline-based LLM's available include:

- Meta's Llama 2 family, and its derivatives
- DataBrick's Free Dolly family
- Technology Innovation Institute's Falcon
- Stanford's Alpaca
- H2O's H2OGPT

Conclusion

In this chapter, we covered...

- Discussed historical path of Generative AI
- Acquired a comprehensive grasp of the essence of Generative AI
- Discussed different types and examples of Generative models
- Highlighted differences between Generative models and Discriminative models

Generative AI Architecture

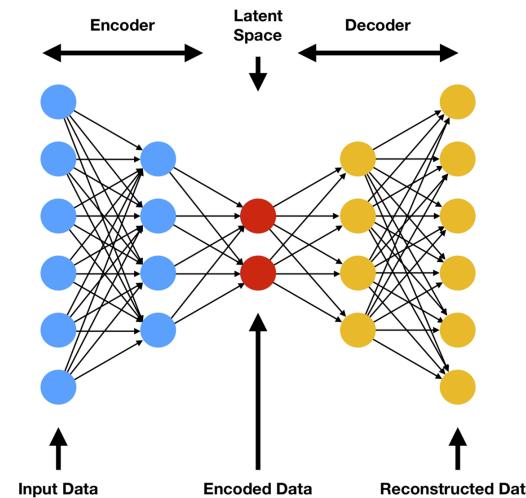
Chapter 2

Agenda

- Variational Autoencoders (VAE)
- Generative Adversarial Networks (GAN)
- Reinforcement Learning from Human Feedback (RLHF)
- Transformers
- Generative Pre-Trained Transformers (GPT)

Variational Autoencoders (VAE)

- Variational Autoencoders (VAEs) are generative models
- VAEs are introduced in 2013 by Kingma and Welling
- They are based on:
 - Autoencoders
 - Variational inference

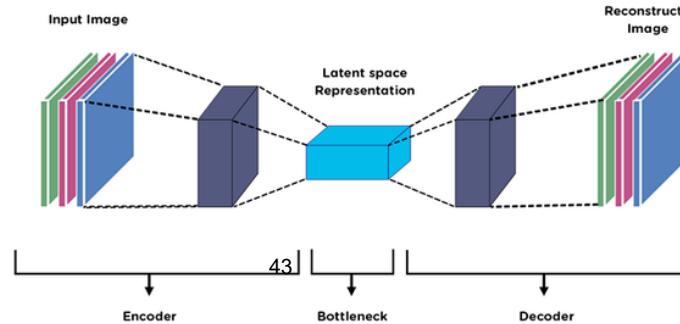


Autoencoder

Autoencoder is a Neural Network which main goal is to reconstruct the input by encoding the input into lower dimensional then decoding it back into a reconstructed output.

It is based on:

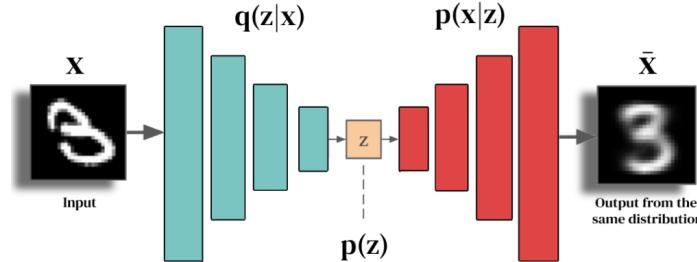
- **Encoder:**
 - The encoder takes the input data and maps it to a lower-dimensional latent space representation.
- **Decoder:**
 - The decoder takes the compressed representation from the **encoder** and reconstructs the original data from it.



Variational Inference

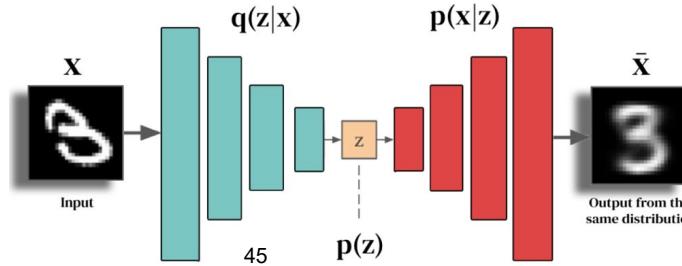
What is Variational Inference?

- **Variational inference** is a Bayesian ML technique
- **It learns parameter distributions instead of point estimates.**
- It is utilized in Variational Autoencoders to model latent variable distributions



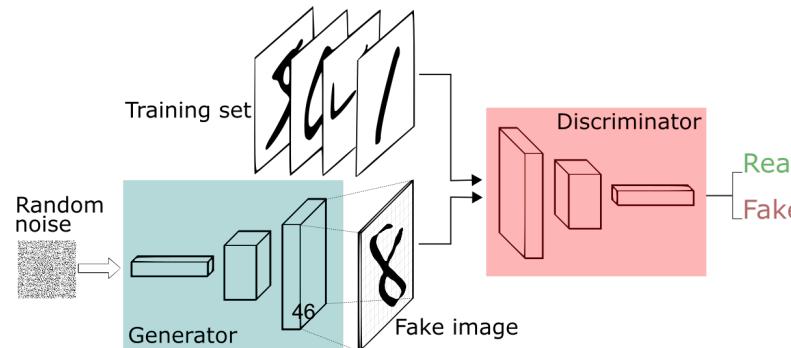
Variational Autoencoders

- VAEs consist of two parts:
 - **Encoder**
 - **Decoder**
- With VAE we don't get a value but instead get a distribution from encoder (parameters of a distribution)
- VAEs are capable of generating samples from the learned distributions in the latent space
- It can perform tasks like image synthesis and data generation.



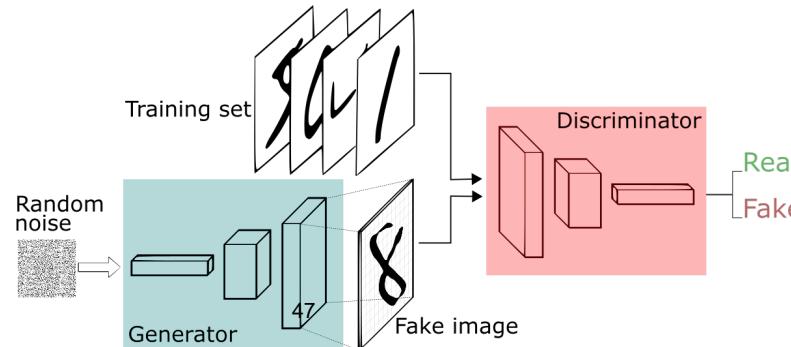
Generative Adversarial Networks (GAN)

- Introduced in **2014** by Ian **Goodfellow**
- Developed to create a new class of generative models
- GANs consist of two neural networks:
 - **Generator** - generates data
 - **Discriminator** - evaluates the authenticity of the data



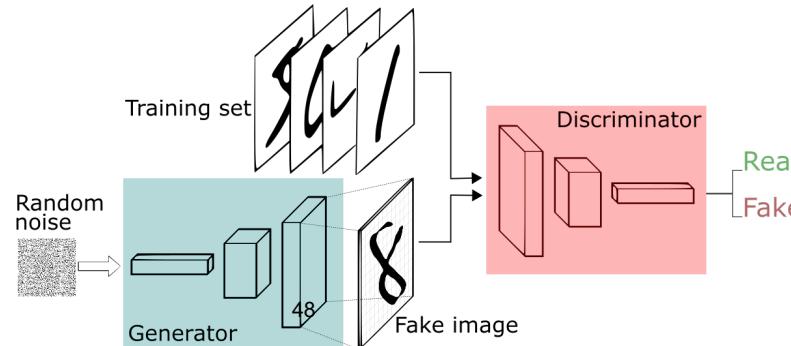
How GANs work

- GANs consist of two neural networks: the **Generator** and the **Discriminator**
 - **The Generator** creates fake data samples from random noise (latent vectors)
 - **The Discriminator** receives both real and fake data and tries to distinguish between them
- They are trained **adversarially**:
 - The **Generator** aims to fool the **Discriminator**
 - The **Discriminator** aims to correctly classify real and fake data



How GANs work

- As training progresses:
 - The **Generator** improves at creating more realistic data
 - The **Discriminator** becomes better at telling real from fake
- The process continues until:
 - The **Generator** produces data that is indistinguishable from real data
 - The **Discriminator** can no longer tell the difference



Applications of GANs

- **Image Synthesis** - generating realistic images from random noise.
- **Video Generation** - creating coherent and diverse video sequences.
- **Style Transfer** - transforming images into different artistic styles.
- **Data Augmentation** - generating additional training data to improve model performance.
- **Anomaly Detection** - identifying unusual patterns in data
- **Text-to-Image Generation** - creating images from textual descriptions.



RLHF

What is RLHF?

- RLHF stands for “**Reinforcement Learning from Human Feedback**”
- RLHF introduces human feedback as an additional source of information for training the AI model.
- Human feedback can come in various forms:
 - **Demonstrations:** Human shows the agent how the task should be performed.
 - **Comparison Feedback:** The agent is given feedback on which of two actions is better
 - **Reward Shaping:** Human experts provide reward signals that guide the agent's learning process

RLHF

Steps of RLHF

- **Pre-training**
 - Start with a pretrained language model
- **Reward Model Training**
 - Train a reward model with human-generated text and rankings
- **Fine-Tuning with RL**
 - Fine-tune the initial model using Proximal Policy Optimization (PPO)
 - PPO is an effective reinforcement learning algorithm

Transformers

- **Transformers:**
 - Deep learning architecture for NLP and generative AI.
- Introduced 2017:
 - By **Vaswani** et al. in "Attention is All You Need."
- Foundation for state-of-the-art NLP models like BERT, GPT-3
- **Self-attention mechanism**
 - The key innovation of transformer models

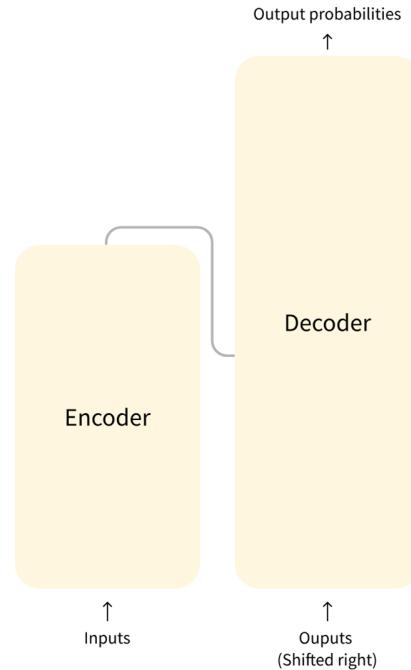
Transformers Overview

Transformers consist of:

Encoder: compresses the input sequence into a mathematical representation

Decoder: generates output for downstream use

- output can be generated text for a use case or as input to another encode/decode layer.



Transformers Overview

Transformers can be stacked in layers

Layers may include encoder/ decoder pairs or just an encoder (as in BERT)

Both encoder and decoder utilize self-attention to derive local context

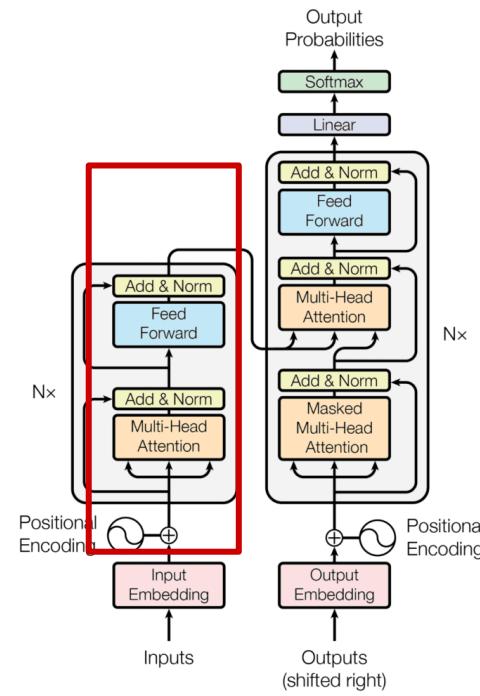
use case: predict next word(s) in a sentence or text string

Transformers - Applications

- **Machine Translation**
 - Outperform traditional sequence-to-sequence models
 - Achieve state-of-the-art performance in language translation.
- **Text Summarization**
 - Transformer models generate coherent and concise summaries of long documents.
 - Valuable for tasks like **news summarization** and **abstractive summarization**.
- **Question Answering**
 - Transformer models understand and answer questions in context
 - Suitable for open-domain question answering
- **Text Generation**
 - Models like GPT-3 generate human-like text from a given prompt or context.
 - Useful for content generation, dialogue systems, and creative writing.

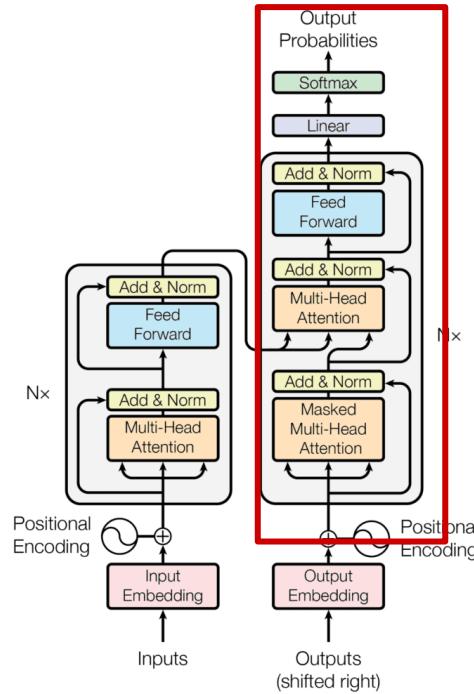
Transformer Architecture

- Encoder
 - The **encoder** receives an input and builds a representation of it (its features).
 - The model is optimized to acquire understanding from the input.



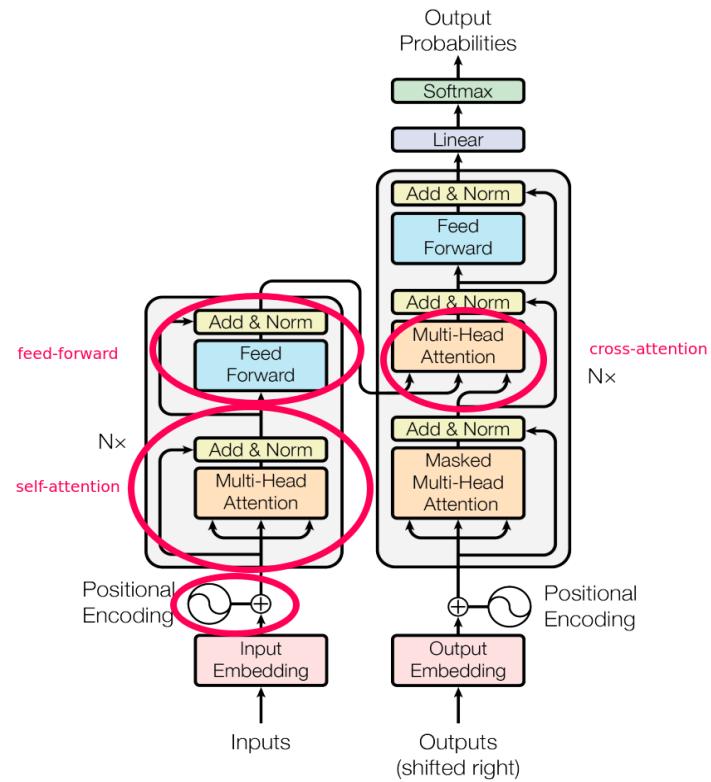
Transformer Architecture

- Decoder
 - The **decoder** uses the encoder's representation (features) along with other inputs to generate a target sequence.
 - The model is optimized for generating outputs.



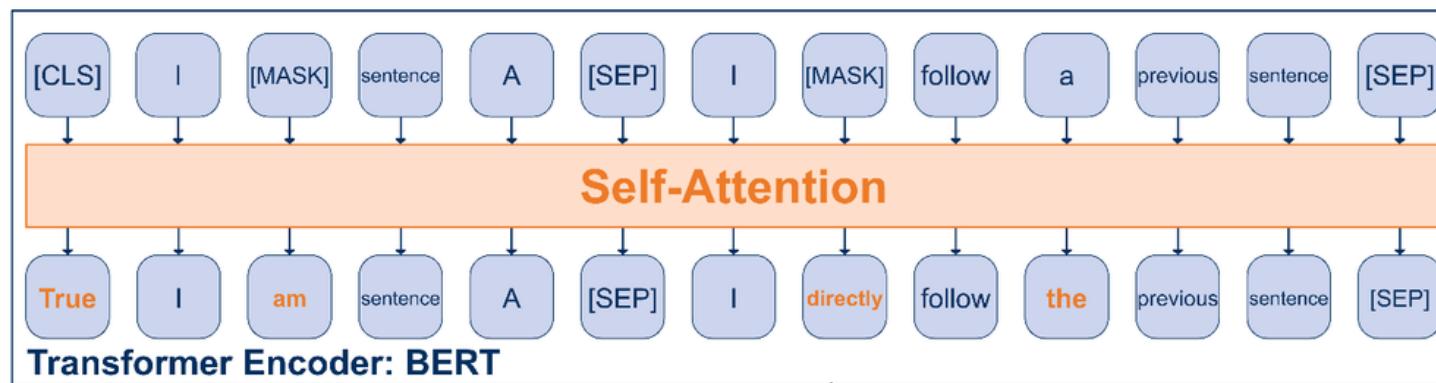
Transformer Architecture

- **Tokenizers**
 - Convert text to tokens and tokens are mapped to embeddings
- **Positional encodings**
 - Inject input word-position information
- **Self-attention layer**
 - Contextually encodes the input sequence information
- **Feed forward layer**
 - Operates bit like a static key-value memory
 - FF layer is similar to self-attention except it does not use softmax and one of the input sequences is a constant.
- **Cross-attention**
 - Decodes output sequence of different inputs and modalities.



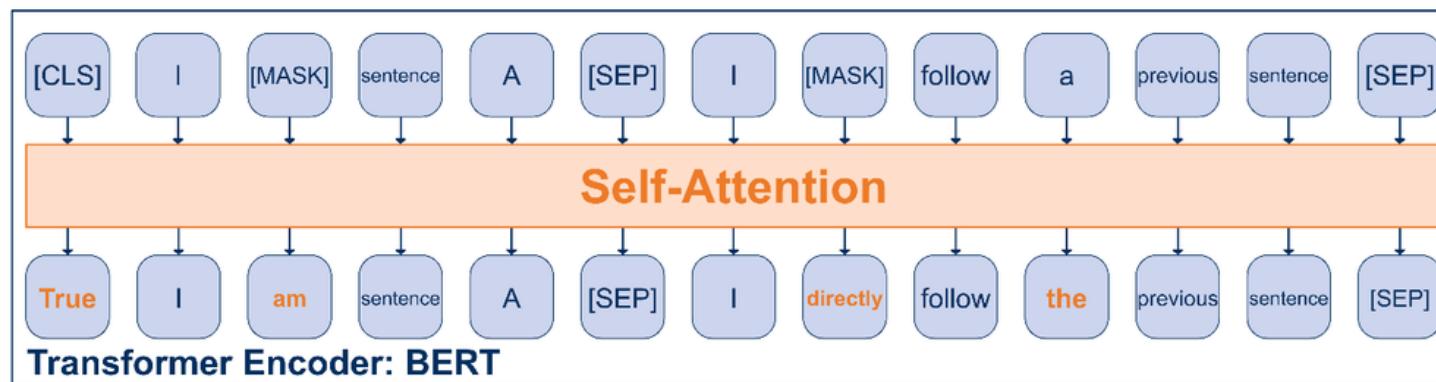
Formatting Data for Training Transformers

1. Take two sequential sentences
2. Apply a mask to random words



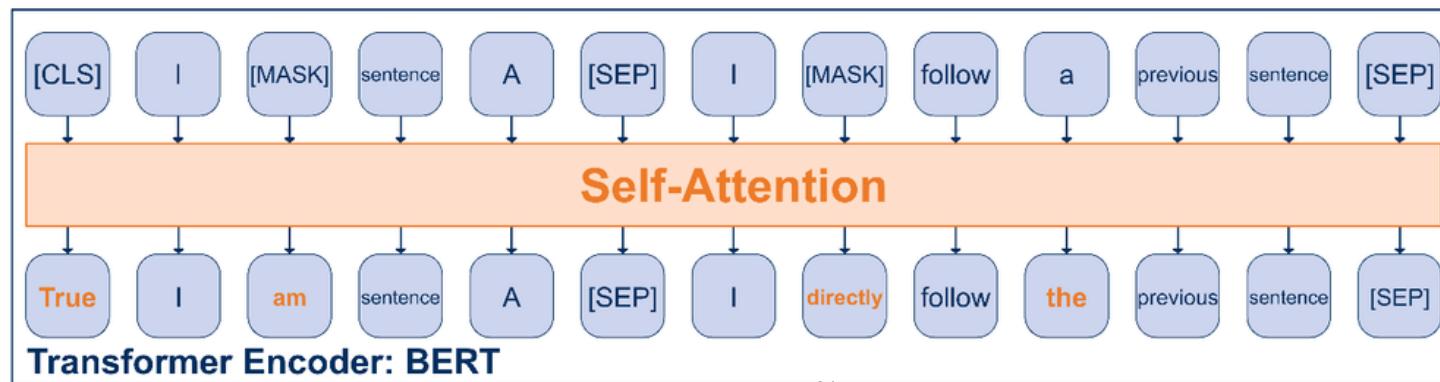
Formatting Data for Training Transformers

- [SEP] - Separator; separates sentences
- [CLS] - Sequence start token, and trained to predict if sentence 2 follows sentence 1
- [MASK] - token for hiding words while training the model



Training Transformers

1. **Next Sentence Prediction (NSP)**: Train to predict whether the second sentence follows the first sentence (train the [CLS] token)
2. **Masked Language Model (MLM)**: Train to predict the word hidden by the [MASK] token



Transformers - Limitations

- **Computational Complexity**
 - The self-attention mechanism has a quadratic complexity with respect to sequence length
- **Model Size**
 - State-of-the-art transformer models have billions of parameters
 - Resource-intensive
 - Difficult to deploy on edge devices
- **Lack of Interpretability**
 - Transformer models are often seen as “black boxes”
 - Challenging to understand their decision-making process.

GPT - Introduction

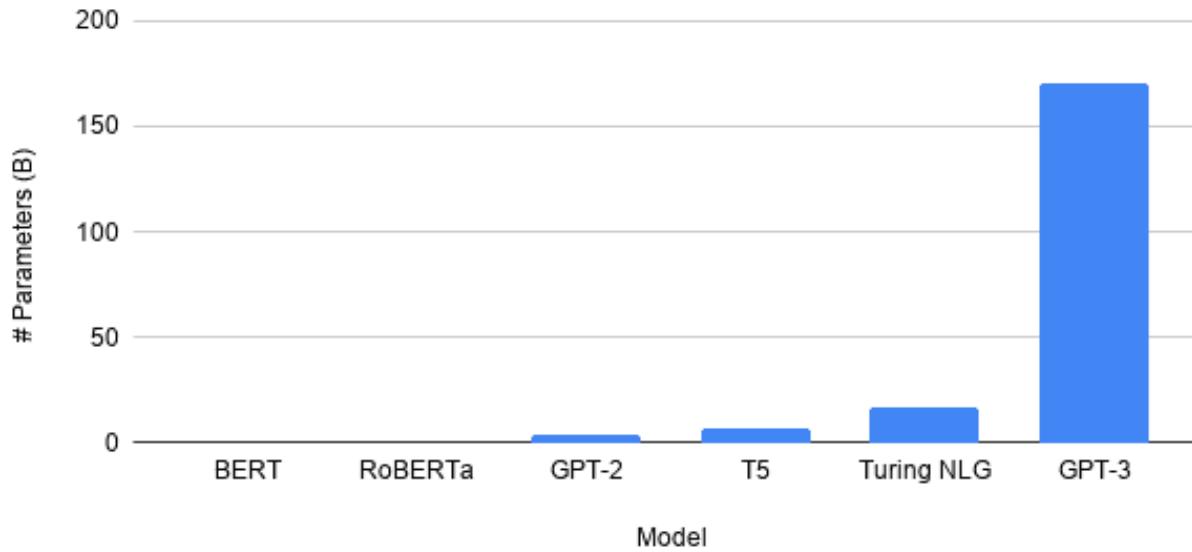
- GPT stands for **Generative Pre-trained Transformer**
- Developed by OpenAI.
- It can:
 - Understand
 - Generate
 - Manipulate human-like text
- GPT has transformed natural language processing and found applications in various fields.



GPT models - History

Model	Architecture	Training Data	Parameters	Release date
GPT-1	12-level, 12-headed Transformer decoder (no encoder)	BookCorpus (4.5 GB of text data)	117 million	June 11, 2018
GPT-2	GPT-1 with modified normalization	40 GB text data, 8 million docs	1.5 billion	February 14, 2019
GPT-3	GPT-2 with modification to allow larger scaling	500 Million tokens (570 GB), WebText, Wikipedia, two books corpora	175 billion	May 28, 2020
GPT-3.5	Undisclosed	Undisclosed	175 billion	March 15, 2022
GPT-4	Trained with both text prediction and RLHF (Reinforcement learning from human feedback). It accepts both text and images as input.	Undisclosed	Undisclosed	March 14, 2023

GPT vs other LLMs



How GPT works?

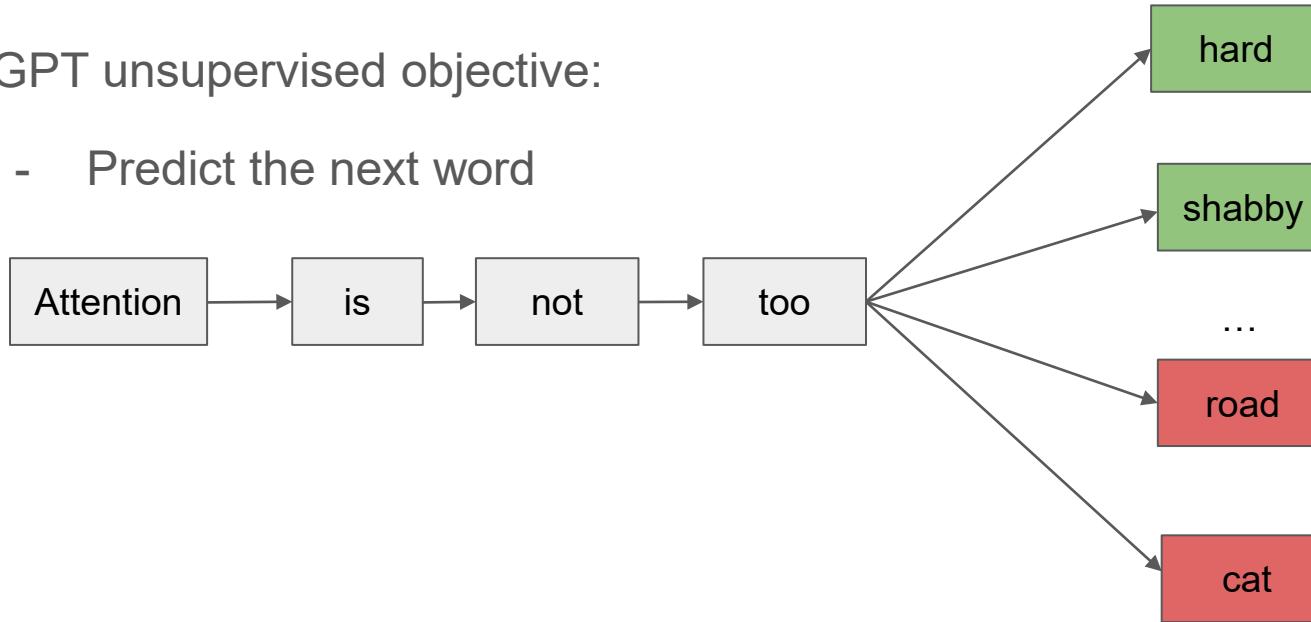
Language Modeling: Predicting the Next Token

- GPT's core task is language modeling
- Transformers use self-attention mechanisms to capture relationships between words in a sentence
- It predicts the probability distribution (e.g., the next sequence of tokens/words), given the previous content.

How GPT works?

GPT unsupervised objective:

- Predict the next word



How GPT is Built?

- GPT is built on the Transformer architecture
- GPT only utilizes the encoder part of the Transformer
- The encoder processes input text sequentially and captures contextual information
- The Transformer architecture is combined with an Autoregressive architecture to improve the Natural Language Generation stage.
- Note: Earlier GPT architectures use the Transformer architecture for the NLG stage, with inferior performance to the combined Transformer/AR architecture of later GPT versions.

How GPT is Trained?

- During pre-training, GPT learns by predicting the next word in a sentence
- A training stage using a GAN architecture removes harmful and false responses
- A training stage involving Reinforcement Learning from Human Feedback (RLHF) improves the quality of responses

GPT - Input/Output

- Input/Output is organized into **tokens**:
 - **Token** is a Numerical representations of words (parts of the word)
 - Numbers are used to represent **tokens**

Tokens	Characters
21	63

What does "Pneumonoultramicroscopicsilicovolcanoconiosis" mean?

Tokens	Characters
21	63

[2061, 857, 366, 47, 25668, 261, 25955, 859, 2500, 1416, 404, 873, 41896, 709, 349, 5171, 36221, 42960, 1, 1612, 30]

Updating GPT model

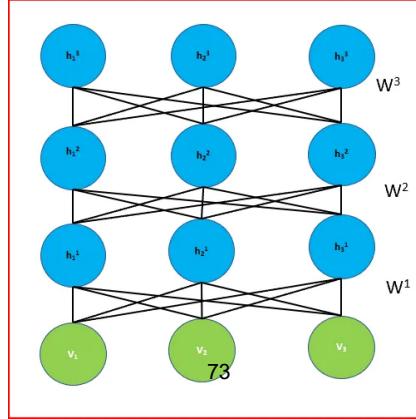
Why does it take long to update GPT model?

- **Complexity of Training**
 - GPT models contains hundreds of billions parameters
- **Data Collection and Annotation**
 - Gathering enough high-quality data
- **Human Feedback and Evaluation (RLHF)**
 - If **RLHF** is used, evaluating model outputs against human feedback is time-consuming
 - This process involves manual work.
- **Compute Resources**
 - It demands substantial computational resources

DELETED/REMOVED

Deep Boltzmann Machines (DBMs)

- DBM is a model with more hidden layers with directionless connections between the nodes
- DBM learns features hierarchically from raw data.
- Extracted features in one layer become hidden variables for the next layer.



Deep Boltzmann Machines - History

Origins in Boltzmann Machines:

- 1980s: Introduction of Boltzmann Machines (BMs), probabilistic models for unsupervised learning.
- Limited success due to training challenges and scalability issues.

Rise of Deep Learning:

- 2000s: Renewed interest in neural networks, leading to advancements in deep learning techniques.
- Deep architectures showed promise in feature learning and representation.

Birth of Deep Boltzmann Machines:

- 2006: Geoffrey Hinton's work on layer-wise training of restricted Boltzmann machines (RBMs).
- Transition from shallow to deep architectures by stacking RBMs, paving the way for Deep Boltzmann Machines.

Deep Boltzmann Machines - Current Status

Challenges and Evolution:

- DBMs faced training difficulties, convergence issues, and high computational demands.
- Researchers explored alternatives like Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs).

Shift in Focus:

- Late 2010s: Research community started favoring VAEs and GANs due to ease of training and better performance.
- DBMs lost prominence but still contributed insights to the understanding of deep learning models.

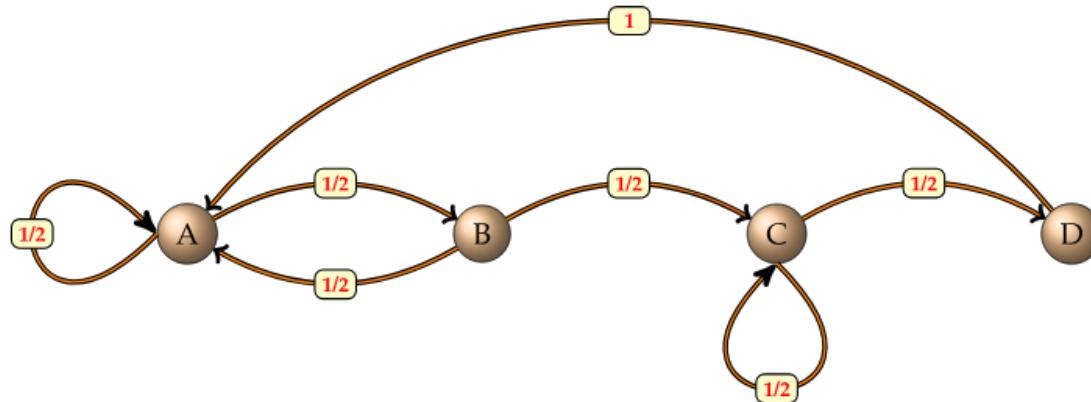
Current Status:

- DBMs are less prevalent in generative AI compared to VAEs and GANs.

Markov Chain Monte Carlo

- **Definition**

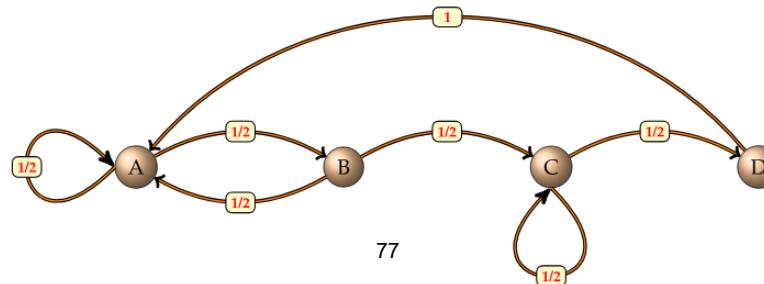
Markov Chains are a mathematical model used to represent a stochastic process, where the future state of the system depends only on the current state and not on the sequence of events that preceded it.



Markov Chain Monte Carlo

Markov Chain can be represented as a **directed graph** where:

- **Nodes** represent states
- **Edges** represent transitions between states
- Each **edge** is assigned a **probability**
 - It indicates the likelihood of state transition from one to another
- The sum of probabilities for all outgoing edges from a state equals 1.



Advantages of Markov Chains in generative AI

- **Simplicity**
 - Markov Chains are relatively simple to implement and understand
- **Scalability**
 - Markov Chains can be easily scaled to handle large datasets and high-dimensional state spaces
- **Flexibility**
 - Markov Chains can be applied to a wide range of data types, including text, music, and images.

Limitations of Markov Chain in generative AI

- **Memorylessness**
 - Due to the Markov property, the model does not consider the history of past states, which may lead to less accurate or coherent generated sequences.
- **Stationary Assumption**
 - Markov Chains assume that the transition probabilities between states are constant over time, which may not hold true for all datasets.
- **Limited Expressiveness**
 - Markov Chains may struggle to capture complex patterns and dependencies in the data.

Applications of Markov Chains in Generative AI

- **Text Generation**
 - It can be used to generate text by modeling the probability of a word or character following another
- **Music Generation**
 - It can be employed to generate music by modeling the probability of a note or chord following another
- **Image Generation**
 - It can be used to generate images by modeling the probability of a pixel value given its neighboring pixel values.

Tuning Generative AI Models

Chapter 3

Agenda

Building Generative AI Models

How Pre-Training Works

Data Preparation and Preprocessing

Fine Tuning Generative AI Models

Formatting Data for LLM Fine Tuning

Fine Tuning GPT

Building Generative AI Models

Building the Pre Trained LLM

- Identify and Collect Training data
- data prep/ETL
- train the predictive model
- fine tune the model using a GAN to minimize harmful results
- fine tune the model further using RLHF to optimize for useful results

Building an Application with the LLM

- Simple LLM queries are “0 shot” predictions
 - They leverage the existing knowledge in the LLM
 - LLM’s can be fine tuned to modify their performance
 - BERT models are typically fine tuned first
 - GPT models typically use prompt engineering or RAG techniques first

Building an Application with the LLM

- Adding context to the prompt creates a “1 shot” predictions
 - App add context to user provided prompts
 - Which turn the queries into “1 shot”
 - Chatbots add the previous conversation thread as context
 - This enables “few shot” queries
 - Prompt templates and Prompt Engineering give a framework for the app’s LLM queries
- Retrieval Agumented Generation (RAG) and Vector Indexing
 - External documents can add addition context to prompts, inducing responses based on specific documents or new data

Collection of data

- **Identify Data Needs**
 - Determine the type of data required for the model's task
- **Data Collection**
 - Gather relevant and diverse data sources for training
- **Domains and Genres**
 - Include various domains, genres and languages
- **Quality Focus**
 - Prioritize data quality over quantity

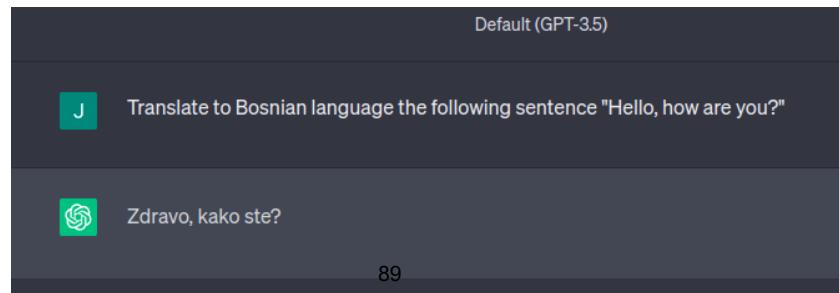
Data Preparation / ETL

- **Data Cleaning**
 - Remove inconsistencies, duplicates, and errors from the collected data.
- **Data Transformation**
 - Convert and preprocess data into a suitable format for training.
- **Data Loading**
 - Organize and structure the prepared data for model training.

LLMs as 0-Shot Learning

LLMs are 0-Shot Predictive models:

- LLMs are designed to predict the next word
- They can generate coherent and contextually relevant text without specific training on a task
- LLMs require no fine-tuning
- They capture semantic and syntactic nuances



LLMs as 1-Shot Learning

1-Shot Prediction with Context

- LLMs can perform 1-shot predictions with context.
- They generate text based on a single input, considering both the prompt and context.
- The context guides them in producing coherent and appropriate text.

Applications:

- Conversational Assistants
- Summarization

LLMs as Multi-Shot Learning

Multi-Shot Context

- Some application sends multiple inputs with context to the model
- Your previous conversion is used as context (example: ChatGPT)

Template Incorporation

- App-level processing includes templates and formatting
- Example: **LangChain** does this for you

Custom Apps

In order to build Custom App based on LLM you can do:

- **Customization**
 - Fine-tune the model for specific app requirement
- **Vector Indexing**
 - Use **vector indexing** to match user inputs with responses
- **Context Enhancement**
 - App adds previous prompts/responses for improved context
- **System Message Integration**
 - Incorporate style, format, and other elements into prompts.

How Pre-Training Works

3.2 Pre-training

The term “**pre-trained models**” refers to models that are trained on large amounts of data to perform a specific task such as:

- Natural language processing
- Image recognition
- Speech recognition

Pre-training phases

- 1. Collection** of large-scale text data
- 2. Tokenization** of text into smaller units
- 3. Model selection**
 - Bert, GPT, XLNet etc.
- 4. Training** a model using unsupervised learning
 - The training process refines model weights to capture language patterns, context and relationships.
- 5. Objective**
 - Learn rich language representations without task-specific labels.

Benefits of pre-training

- **Generalization**
 - Pre-training allows models to capture broad language knowledge, enabling them to excel in various downstream tasks.
- **Transfer Learning**
 - Pre-trained LLMs can be fine-tuned for specific tasks with minimal data and training time, making them highly efficient.
- **Data Efficiency**
 - Pre-training on vast corpora helps models comprehend nuances and context, reducing the need for large task-specific datasets.
- **Resource Savings**
 - Training from scratch requires massive computational power and time, while pre-training leverages existing models and infrastructure.
- **Adaptability**
 - Pre-trained LLMs can be used across different languages and domains, promoting cross-lingual and multi-task learning.

Challenges in pre-training Language Models

- **Data Scale and Quality**
 - Pre-training LLMs demands vast amounts of high-quality text data, which can be challenging to collect and process.
- **Computational Resources**
 - Training large-scale LLMs requires significant computational power and memory, making it computationally expensive.
- **Training Time**
 - Pre-training LLMs can take several days or even weeks, affecting the overall development cycle.
- **Generalization**
 - Ensuring that pre-trained models generalize well to various downstream tasks is a non-trivial task.

Fine Tuning over Pre-Training

- Because of the constraints of building new, large models, pre-trained models built by dedicated teams are typically leveraged
- Pre-Trained models can be fine tuned to adapt them to new domains
- note: In some cases, Prompt engineering and other techniques like Vector Indexing can be an alternative to fine tuning

Fine Tuning Pre-Trained Models

Pre-trained models could be used for **fine-tuning**:

1. **Fine-tune** pre-trained model on domain data
2. **How?**
 - **Unfreeze** the classifier
 - **Retrain** it on new data

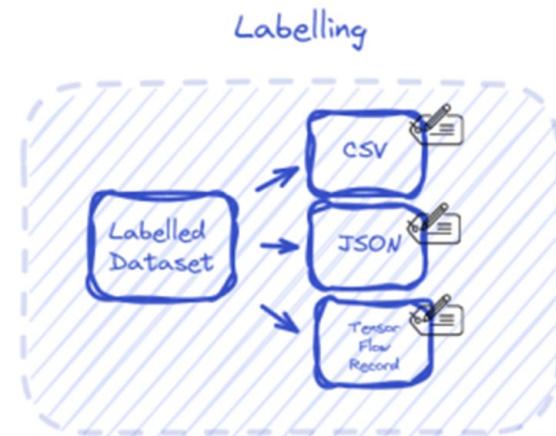
Example:

- Use your labeled sentences for sentiment analysis from specific domain (finance, marketing etc) to improve general sentiment classifier.

Data Preparation and Preprocessing

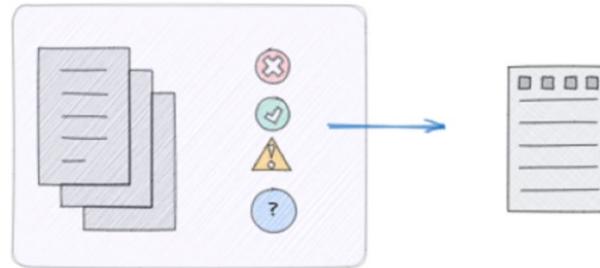
Introduction to Data Preparation and Preprocessing

- Data preparation and preprocessing are critical stages in building effective generative AI models
- Data preparation involves:
 - Collecting
 - Cleaning
 - Preparation and structuring the data



Introduction to Data Preparation and Preprocessing

- Preprocessing focuses on transforming the data into a format suitable for fine-tuning the model
- Proper preprocessing improves:
 - **Model efficiency**
 - **Convergence**
 - **Generalization**



Data Quality and Diversity

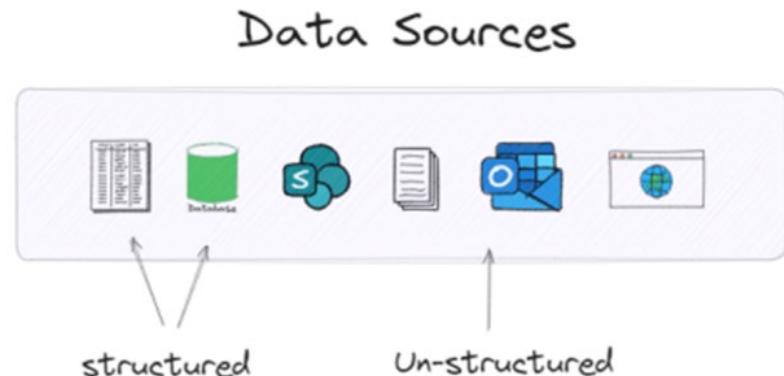
- **High-quality data is the foundation** of successful generative AI models.
- Quality data enables the model to learn:
 - Patterns
 - Semantics
 - Context more effectively
- Good data mitigates:
 - Biases
 - Reduces the risk of generating harmful or misleading outputs

Challenges in Data Preparation

- **Data Noise**
 - Cleaning and filtering text data to remove noise, special characters, and irrelevant information
- **Ambiguity and Inconsistency**
 - Text data often contains ambiguous language
- **Vocabulary Size**
 - Building a suitable vocabulary for tokenization becomes complex due to the vast number of words and subwords in natural language.
- **Out-of-Vocabulary Tokens**
 - Handling OOV tokens during tokenization is critical to ensure the model can process unseen words effectively.

Data Collection

- Common Data sources:
 - Databases (NoSQL, SQL etc)
 - Document parsing
 - Email extraction
 - Web scraping
 - Social media



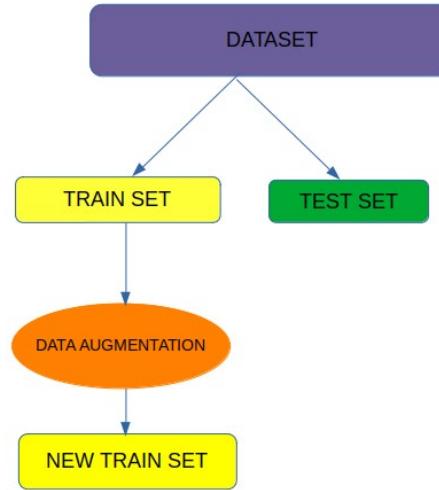
Data Cleaning Steps

Common cleaning steps include:

- **Removing noise**
 - HTML Tags and Markup
 - URLs and Email Addresses
 - Remove unwanted noise present in the images
- **Remove irrelevant text**
 - Sections or documents that are not related to the task or domain of interest
- **Handling special characters and symbols**
 - Copyright and Trademark Symbols ©, ®, ™
 - Emoticons and Emoji
 - Arrows: Symbols like →, ←, ↑, ↓, ↕
- **Resizing and Standardization**
 - Resize images or signals to a uniform size
- **Filtering (for signal data)**
 - Apply filtering techniques like low-pass or high-pass filtering

Augmentation

- Data Augmentation refers to the process of artificially increasing the size and diversity of the training dataset
- Augmentation can be done on:
 - **Images**
 - **Text**
 - **Signals**



Augmentation - Text data

Common **text** data augmentation methods:

- **Paraphrasing**
 - Reword sentences to convey the same meaning using different phrasing or expressions.
- **Back-Translation**
 - Translate the text into another language and then translate it back to the original language
- **Word Dropout**
 - Randomly remove words from the text during training to make model more robust.
- **Random Insertion**
 - Introduce random words or phrases into the text at different positions.
- **Sentence Splitting and Combination**
 - Split long sentences into shorter ones or combine shorter sentences to create new sentence structures.

Augmentation - Image data

Common **image** data augmentation methods:

- **Rotation**
- **Flipping**
 - Creating mirror images of the original image by flipping it horizontally or vertically
- **Translation**
 - Shifting an image along the horizontal and vertical axes by a certain number of pixels
- **Changes in brightness**
- **Changes in contrast**
- **Cropping**
- **Zooming**

Augmentation - Signal data

Common **signal** data augmentation methods:

- **Time-shifting**
- **Adding random noise**
- **Pitch shifting**
 - In audio data, pitch shifting involves changing the pitch (frequency) of the signal while preserving its duration
- **Time stretching and compression**
 - Time stretching involves slowing down or stretching the signal's time axis
- **Time domain filtering**
 - Time domain filtering involves applying filters, such as low-pass or high-pass filters, to the signal.
- **Amplitude scaling**
 - Amplitude scaling adjusts the signal's amplitude¹⁰, making it louder or quieter

Fine Tuning Generative AI Models

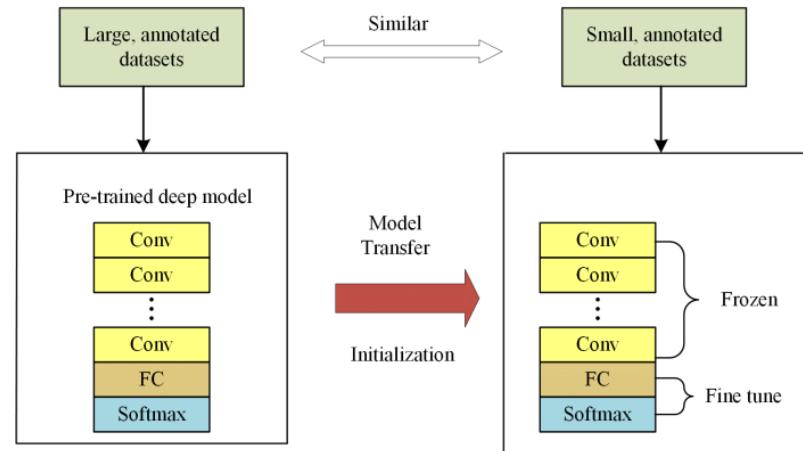
2.3 Fine-tuning

Introduction

- Fine-tuning is a transfer learning technique where **a pre-trained neural network model is further trained**
- It is done on a new dataset
- Fine-tuning can **help to achieve higher accuracy** with less training data

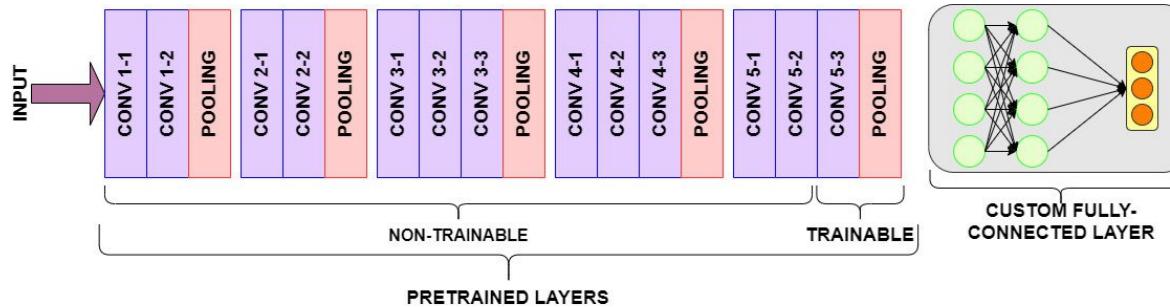
Why to use Fine-tuning?

- Building and validating ML model from scratch is:
 - Time consuming
 - Expensive
- **What if we can find a trained model that already does one task well similar to ours?**
 - Fine-tuning helps to solve this issue



When to use Fine-tuning?

- Lack of data for specific task
- Lack of sufficient computational power
- The new task is similar to the original task



Fine-tuning procedure

1. Pre-training

- Example: GPT-3 model by OpenAI was pre-trained using a dataset of 45TB of text.

2. Task-relevant layers

- Add extra layers to modify the learned representations
- Example: The task-specific layers for sentiment analysis

3. Data Preparation

- Example: Gather sentences with positive/negative/neutral labels for sentiment analysis

4. Fine-tuning

- Adapting the pre-trained model's learned representations to the target task by training it on task-specific data

5. Iteration and Evaluation

Fine-tuning models - examples

OpenAI's base models are suitable for Fine-tuning:

- Davinci
- Curie
- Babbage
- Ada



GPT-3 base models and their features

GPT-3

GPT-3 models can understand and generate natural language. These models were superceded by the more powerful GPT-3.5 generation models. However, the original GPT-3 base models (`davinci` , `curie` , `ada` , and `babbage`) are current the only models that are available to fine-tune.

LATEST MODEL	DESCRIPTION	MAX TOKENS	TRAINING DATA
<code>text-curie-001</code>	Very capable, faster and lower cost than Davinci.	2,049 tokens	Up to Oct 2019
<code>text-babbage-001</code>	Capable of straightforward tasks, very fast, and lower cost.	2,049 tokens	Up to Oct 2019
<code>text-ada-001</code>	Capable of very simple tasks, usually the fastest model in the GPT-3 series, and lowest cost.	2,049 tokens	Up to Oct 2019
<code>davinci</code>	Most capable GPT-3 model. Can do any task the other models can do, often with higher quality.	2,049 tokens	Up to Oct 2019
<code>curie</code>	Very capable, but faster and lower cost than Davinci.	2,049 tokens	Up to Oct 2019
<code>babbage</code>	Capable of straightforward tasks, very fast, and lower cost.	2,049 tokens	Up to Oct 2019
<code>ada</code>	Capable of very simple tasks, usually the fastest model in the GPT-3 series, and lowest cost.	2,049 tokens	Up to Oct 2019

Formatting Data for LLM Fine Tuning

Formatting Data for Text Classification

- **Prompt:** “Lionel Messi scores hat-trick as Barcelona win against Real Madrid.”
 - **Completion:** “Sports”
- **Prompt:** “Donald Trump announces new tariffs on Chinese imports.”
 - **Completion:** “Politics”
- **Prompt:** “Taylor Swift wins Album of the Year at the Grammys.”
 - **Completion:** “Entertainment”

Formatting data for Language generation

- **Prompt:** “Please write a product description for a portable blender.”
 - **Completion:** “This sleek and compact blender is perfect for on-the-go smoothies and shakes. Its powerful motor and durable blades make it easy to blend even though ingredients, while its lightweight design and rechargeable battery make it ideal for travel and outdoor adventures.”
- **Prompt:** “Please write a product description for a smartwatch.”
 - **Completion:** “Stay connected and organized with our stylish and feature-packed smartwatch. With its intuitive touch screen and advanced health monitoring capabilities, this watch is the perfect companion for busy professionals and fitness enthusiasts. Its long-lasting battery and water-resistant design make it a reliable choice for any activity.”

OpenAI input format

```
{"prompt": "<prompt text>", "completion": "<ideal generated text>"}
```

```
{"prompt": "<prompt text>", "completion": "<ideal generated text>"}
```

```
{"prompt": "<prompt text>", "completion": "<ideal generated text>"}
```

...

Selecting Appropriate Datasets

Data Quality:

- **Quality Over Quantity**
 - Prioritize high-quality data over a large quantity of data.
 - Accurate and well-labeled examples are more valuable than a massive dataset with noise.
 - Ensure that the data is representative of the specific task or domain you're targeting.

Data Diversity

- **Annotate Diverse Examples**
 - Annotate examples that cover a wide range of variations, contexts, and entity types relevant to your task.

Domain-specific data

Your training data should be:

- **Large**
 - Ideally thousands or tens of thousands of examples
- **High-quality**
 - Consistently formatted and cleaned of incomplete or incorrect examples
- **Representative**
 - Training data should be similar to the data upon which you'll use your model
- **Sufficiently specified**
 - Containing enough information in the input to generate what you want to see in the output

Fine Tuning GPT

Steps to Fine-Tune ChatGPT

1. Define the Use Case
2. Collect and Preprocess Data
3. Prepare Data for Training
4. Fine-Tune the Model
5. Evaluate the Model
6. Deploy the Model



Steps to Fine-Tune ChatGPT

- Define the Use Case
 - Collect and Preprocess Data
 - Prepare Data for Training
 - Fine-Tune the Model
 - Evaluate the Model
 - Deploy the Model
- Use case identification is the **initial step** for optimizing ChatGPT.
 - **Clear understanding** of your goal enhances ChatGPT's effectiveness.
 - ChatGPT customization suits multiple purposes
 - sentiment analysis
 - language translation
 - question answering
 - etc.



Steps to Fine-Tune ChatGPT

- Define the Use Case
 - Collect and Preprocess Data
 - Prepare Data for Training
 - Fine-Tune the Model
 - Evaluate the Model
 - Deploy the Model
- Fine-tuning ChatGPT relies heavily on the quality and quantity of the data
 - For the intended purpose, the data must be:
 - Accurate
 - Suitable



Steps to Fine-Tune ChatGPT

- Define the Use Case
 - Collect and Preprocess Data
 - Prepare Data for Training
 - Fine-Tune the Model
 - Evaluate the Model
 - Deploy the Model
- Data is divided into:
 - Training
 - Validation
 - Testing



Steps to Fine-Tune ChatGPT

- Define the Use Case
 - Collect and Preprocess Data
 - Prepare Data for Training
 - **Fine-Tune the Model**
 - Evaluate the Model
 - Deploy the Model
- After data cleaning and organization, the model is fine-tuned.
 - Initial step involves training the **pre-trained ChatGPT** on specific data.
 - Available models:
 - Ada
 - Babbage
 - Curie
 - Davinci
 - GPT-3.5
 - GPT-3.5 16k
 - GPT-4
 - GPT-4 32k



Steps to Fine-Tune ChatGPT

- Define the Use Case
 - Collect and Preprocess Data
 - Prepare Data for Training
 - Fine-Tune the Model
 - Evaluate the Model
 - Deploy the Model
- Model performance is evaluated on the **test set**.
 - Assessment metrics include:
 - Accuracy
 - Precision
 - Recall
 - F1 score
 - Perplexity
 - FID
 - ROUGE
 - ...



Steps to Fine-Tune ChatGPT

- Define the Use Case
- Collect and Preprocess Data
- Prepare Data for Training
- Fine-Tune the Model
- Evaluate the Model
- Deploy the Model
- Deployment is the final stage after performance validation
- Creating a RESTful API or web app enables user interaction with the model.



How to set up and fine tune ChatGPT

1. Create OpenAI Key
2. Installing the OpenAI library
3. Prepare data
4. Fine-tune ChatGPT
5. Deploy the model



Fine-Tune ChatGPT

```
import openai
import json
import time

# Set your API key
openai.api_key = "YOUR_API_KEY"

# Load your training data
with open("training_data.json", "r") as f:
    training_data = json.load(f)

# Initialize a new model with the base model you want to fine-tune
model = "text-davinci-002"
fine_tuned_model = openai.Model.create(
    model=model,
    fine_tune=True,
    training_data=training_data
)

# Wait for the model to finish fine-tuning
while fine_tuned_model.status()["data"]["ready"] is False:
    time.sleep(30)
fine_tuned_model = openai.Model.retrieve(fine_tuned_model.id)
print("Fine-tuning complete")
```



Deploy the model

```
# set your API key
import openai
# Create a new model object using the ID of the fine-tuned model
model_id = "YOUR_FINE_TUNED_MODEL_ID"
model = openai.Model(model_id)
# Make an API request to the model to generate text
prompt = "Prompt ->"
response = model.generate(prompt)
# Print the response
print(response.choices[0].text)
```



Case Studies and Real-World Applications

Chapter 4

Agenda

- Design Considerations
- Generative AI for Text
- Generative AI for Media
- Generative AI for Code

Design Considerations

Design Considerations

When designing a GenAI application, consider the following requirements when choosing an LLM:

- **Input/Output Size**
- **Security/Privacy**
- **Performance and Scaling**
- **Cost**
- **Features**

Designing for Input/Output Sizes

- LLM's have a maximum context, specified as the max number of tokens
- This includes both the input and the output together
- Estimate 0.3 words/token
- Long chunks of text can be broken up into smaller chunks, and processed sequentially

Input/Output Sizes

- LLM's have a maximum context, specified as the max number of tokens
- This includes both the input and the output together
- Estimate 0.3 words/token
- Long chunks of text can be broken up into smaller chunks, and processed sequentially or hierarchically
- Example: a long summarization turned into a sequence of summarizations, then the summarizations summarized

Security and Privacy

- Consider the sensitivity and regulatory requirements of the data
- Use offline LLM's when working with highly sensitive data, or when an online LLM is not an option
- Offline LLM's will have hardware and resource requirements, often requiring GPU's and enough GPU RAM

Performance & Scaling

- Consider the capabilities of the LLM against the application's requirements
- Larger LLM's will perform better on NLP tasks, but may be slower to generate responses
- The speed of responses will be slower for offline LLM's
- Offline LLM's do not scale
- Consider fine tuning requirements in this design as well

Costs

- Online LLM's charge per API call or per tokens generated
- Initially, this is cheaper, but can add up over time
- Offline LLM's have a significant infrastructure cost (either actual servers, or cloud resources)
- Initially this is more expensive, but is a fixed cost (until needing to scale)
- Consider the fine tuning costs as well

Features

- Consider the tasks the LLM is best for against the application requirements
- Larger LLM's are not always better - smaller ones can perform better on some types of tasks
- There may be options within the same family, for example: GPT3 Currie vs Davinci, or Free Dolly -3b, -7b, and -12b
- Consider the fine tuning API interface as well

Generative AI for Text

Generative Text App Examples

Example Applications:

- Copy.ai for writing
- Jasper for marketing
- Bing for search + GenAI LLM
- ChatGPT

Copy AI

copy.ai

2023-10-05 Untitled New Project

Chat My Projects Infobase Brand Voice Workflows >

Welcome to Chat by Copy.ai

Get started by writing a task and Chat can do the rest. Not sure where to start? Check out the Prompt Library for inspiration.

Real-Time Search

"Summarize the latest news on generative AI!"
"Write a personalized email to [insert LinkedIn profile URL]"

Editor

Long Form Content

"Create a blog post about search engine optimization"
"Write a press release about www.copy.ai"

Brainstorm Ideas

Ask or search anything

Upgrade to Pro

Tricia's Workspace

Browse Prompts No Brand Voice Improve

Copy AI

A Letter of Love for My Husband

Clear Chat

Prompts

Try "Sales" or "Email"

Custom

Content/SEO

Email Marketing

Paid Ads

PR/Communications

Recruiting

Sales

Social Media

Strategy

Attention-Interest-Desire-Action

Email Subject Generator

Features-Advantages-Benefits

Newsletter From Recent News

Newsletter Inspiration

Pain-Agitate-Solution

PREVIEW (WORKSPACE PROMPT) ⓘ

Using the 'Attention-Interest-Desire-Action' framework, write an email marketing campaign that highlights the [features] of our [product/service] and explains how these [advantages] can be helpful to [ideal customer persona]. Elaborate on the [benefits] of our product and how it can positively impact the reader.

148

Use Prompt

This screenshot shows the Copy AI interface with a workspace prompt for an email marketing campaign. The left sidebar lists various categories like Custom, Content/SEO, Email Marketing, Paid Ads, etc. The 'Email Marketing' category is selected. A preview window on the right displays the 'Attention-Interest-Desire-Action' framework, which involves highlighting features, advantages, and benefits, explaining their value to the ideal customer persona, and elaborating on the benefits for the reader. The preview text is as follows:

Using the 'Attention-Interest-Desire-Action' framework, write an email marketing campaign that highlights the [features] of our [product/service] and explains how these [advantages] can be helpful to [ideal customer persona]. Elaborate on the [benefits] of our product and how it can positively impact the reader.

Copy AI

- Generate blog posts and other copy
- Uses Prompt Templates
- Uses Prompt Engineering for context
- Chat feature adds chat conversation as context to each subsequent prompt API call

JASPER AI

- Built **on top of** custom, **proprietary AI language models** and other third-party models like Cohere, OpenAI, and Anthropic
- Fine tuned for **marketing content**
- Trained by reading approximately 10% of published information online



JASPER AI

The screenshot shows the Jasper AI web interface. On the left is a sidebar with navigation links: Dashboard, **Templates** (which is selected), Recipes, Documents, Trash, AI outputs, Favorites, and another Trash link. Below this is a 'PROJECT' dropdown set to 'Personal'. At the bottom left is a user profile icon for 'Blogging Ocean'.

The main area is titled 'Templates' with a search bar at the top. A navigation bar below the title includes 'All' (selected) and categories: Frameworks, Email, Website, Blog, Ads, Ecommerce, Social Media, New, Google, Video, SEO.

The template cards are:

- Documents**: 'BOSS MODE' icon. Description: Let Jasper help you write longer articles from start to finish.
- Paragraph Generator**: Document icon. Description: Create paragraphs that will captivate your readers.
- AIDA Framework**: AIDA icon. Description: Use the oldest marketing framework in the world. Attention, Interest, Desire, Action.
- PAS Framework**: PAS icon. Description: Problem-Agitate-Solution. A valuable framework for creating new marketing copy ideas.
- Content Improver**: Pencil icon. Description: Take a piece of content and rewrite it to make it more interesting, creative, and engaging.
- Product Description**: Cloud icon. Description: Create compelling product descriptions to be used on websites, emails and social media.
- Blog Post Topic Ideas**: Chat icon. Description: Brainstorm new blog post topics that will engage readers and rank well on Google.
- Blog Post Outline**: Chat icon. Description: Create lists and outlines for articles. Works best for "Listicle" and "How to" style blog posts or articles.
- Blog Post Intro Paragraph**: Chat icon. Description: Blast through writers block by letting Jasper write your opening paragraph. Updated 2d ago.
- Blog Post Conclusion Paragraph**: Checkered flag icon. Description: Wrap up our blog posts with an engaging conclusion paragraph.

JASPER AI

 **One-Shot Blog Post** ☆

Generate a full blog post with intro, body, and conclusion.

Blog Topic
SEO, Social Media 0/1500

 [Knowledge base](#) ▾

Tone of voice ⓘ
Professional 0/100

Intended Audience
CMO's, CIO's 0/100

Beta [Language options](#) ⓘ
DeepL integration is currently disabled. [Enable in Settings](#).

Input language English → **Output language** English (American) **Formality** Default

Clear inputs 1 152 Generate content →

JASPER AI

- Combines multiple user inputs
- Constructs the Prompt API call with these inputs
- Uses Prompt Templates to construct calls
- Based on OpenAI's GPT3 LLM
- Also provides an auto-complete feature, that uses Prompt Engineering to add context to each prompt API call.

Bing

The screenshot shows the Microsoft Bing homepage with the following elements:

- Header:** Microsoft Bing, SEARCH, CHAT (underlined), Sign in, 15 notifications, and a menu icon.
- Bing Logo:** The classic blue 'b' logo.
- Welcome Message:** "Welcome to the new Bing" and "Your AI-powered copilot for the web".
- Call-to-action Buttons:**
 - Ask complex questions:** "What are some meals I can make for my picky toddler who only eats orange-colored food?"
 - Get better answers:** "What are the pros and cons of the top 3 selling pet vacuums?"
 - Get creative inspiration:** "Write a haiku about crocodiles in outer space in the voice of a pirate"
- Input Field:** "Ask me anything..." with a microphone icon.
- Feedback:** "Feedback" button with a blue checkmark.
- Page Footer:** "Waiting for www.bing.com" and page number "154".

Bing

- Incorporates Chat style responses
- Also provides links to source material
 - Done using RAG
- Uses OpenAI GPT as underlying LLM

Bing

Microsoft Bing SEARCH CHAT

Sign in 10 🔍 ⚙️

Abraham Lincoln was born in 1809 to an illiterate father and a mother who only had basic reading skills. He himself did not receive extensive formal education and only attended school occasionally.¹ According to Lincoln himself, he only attended school every now and then and described his formal education as "flawed".¹ When he was nominated for the presidency of the United States in 1860, he felt sad about his meager formal schooling.¹ His formal education amounted to only a year.¹ However, he was self-educated and sustained his appetite for independent learning all his life. Lincoln's stepmother, Sarah, described him as a boy who was always reading something and had a fascination with learning. Even as an adult, Lincoln continued to read to improve himself as a person, studying books on English grammar and immersing himself in the Greek mathematician Euclid when he served in Congress.¹

Learn more: 1. [totallyhistory.com](#) 2. [en.wikipedia.org](#) 3. [en.wikipedia.org](#) +2 more 1 of 4

What was Lincoln's profession before he became president? How did Abraham Lincoln teach himself to read and write?

New topic Ask me anything... 0/2000 ➤ Feedback

156

Bing

- Based on GPT
- Uses Prompt Engineering to pass session/conversation as context to LLM API call
- Uses RAG to combine Search Engine with ChatBot, and return sources along with the text explanation

ChatGPT

- Offers common users **free access to basic AI content**
- Key Features:
 - Natural Language Understanding
 - Conversational Context
 - Open-Domain Conversations
 - Language Fluency



ChatGPT



ChatGPT **PLUS**

Explain options trading
if I'm familiar with buying and selling stocks

Tell me a fun fact
about the Roman Empire

Plan a trip
to explore the rock formations in Cappadocia, Turkey

Come up with concepts
for a retro-style arcade game

Send a message



ChatGPT

- Chatbot style question and answer
- Uses OpenAI's GPT3 (free version) or GPT4 (paid version) as underlying LLM
- Uses Prompt Engineering to pass session/conversation as context to LLM API call
- New feature “Custom Instructions” uses prompt templates to add context/instructions to each new prompt

Generative AI for Media

Image Synthesis

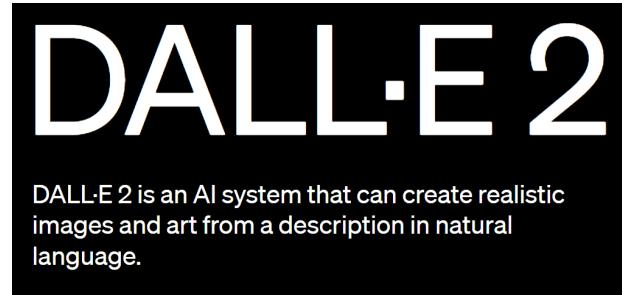
- Image Synthesis is about creating new images from scratch or transforming existing ones.
- Examples include DALL-E by OpenAI which creates images from text descriptions, and StyleGAN, a model by Nvidia capable of creating highly realistic human faces.

Image Synthesis

- Techniques:
 - Generative Adversarial Networks (GANs): Models that can generate novel, realistic images by learning from a dataset of existing images.
 - Variational Autoencoders (VAEs): Models that can generate new images by learning the underlying distribution of training data.
- Applications:
 - Art and Design: AI can generate unique artwork and design elements.
 - Advertising: AI can create eye-catching and unique visual content for ads.
 - Medical Imaging: AI can help visualize medical conditions or predict patient outcomes.

DALLE

- From OpenAI
- Generates images based on text prompts
- Application uses prompt templates to select different styles of art
- This is a nascent technology, not as developed as LLM's
- Has issues with mistakes in images, including some common mistakes:
 - Hands and feet in wrong place, or not to scale
 - Objects floating or not in a realistic location
- Subsequent prompts can refine these errors, but not completely



DALLE

The screenshot shows the DALL-E interface. At the top, there is a navigation bar with the DALL-E logo, the word "DALL-E", and links for "History" and "Collections". On the right side of the top bar are three dots and a profile icon. Below the navigation bar is a large, empty white area where generated images would appear. In the center-left, there is a text input field containing the placeholder text "An Impressionist oil painting of sunflowers in a purple vase...". To the right of this input field is a "Generate" button. Above the input field, there is a smaller text input field with the placeholder "Start with a detailed description" and a "Surprise me" button. Below the main input field, there is a link "Or, upload an image to edit". At the bottom left, there is a small icon of a credit card with the number "0" next to it, and a "Buy credits" button. At the very bottom, there is a link "Learn more about credits".

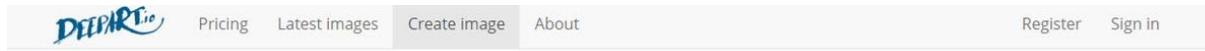
Image Synthesis

- Opportunities:
 - Custom Art Creation: Artists can use AI to generate unique pieces of art.
 - Image Enhancement: AI can enhance low-quality images.
- Challenges:
 - High computational requirements: Generating high-quality images requires significant computational power.
 - Avoiding illegal or harmful image generation: Ensuring AI doesn't generate offensive, inappropriate, or copyrighted images.

Image Synthesis

- Future Prospects:
 - With advancements in technology, AI could create high-resolution, photorealistic images for any given text description.
 - AI could be used to create custom designs for everything from furniture to clothing.

Image Synthesis



PHOTO

Upload photo

Drop your photo here or
click to select one from your computer.

STYLE

Popular Upload Existing



e-mail

e-mail

I accept privacy policy

Submit

Music Composition

- AI can create new music pieces or enhance existing ones by learning from a dataset of music.
- Examples include MuseNet by OpenAI, capable of composing music with 10 different instruments, and Jukin Media's Jukin Composer, which generates royalty-free music for videos.

Music Composition

- MuseNet by OpenAI



A screenshot of the MuseNet web application. At the top, there are dropdown menus for 'STYLE' (set to 'CHOPIN'), 'INTRO' (set to 'NONE'), and 'INSTRUMENTS' (listing PIANO, STRINGS, WINDS, DRUMS, HARP, GUITAR, and BASS). A slider for 'NUMBER OF TOKENS' is set to 50. Below this, a 'HIDE ADVANCED SETTINGS' button is visible. The main area features a piano-roll style visualization of musical notes over three staves. At the bottom, there is a play button labeled 'PLAY FROM START', download and tweet buttons, and a 'RESET' button. A note at the bottom states: 'Some of MuseNet's limitations include: 170. The instruments you ask for are strong suggestions, not requirements. MuseNet'.

Music Composition

- Techniques:
 - LSTM: Used for sequence prediction problems, such as predicting the next note in a melody.
 - Transformers: Used to understand relationships between different elements of a piece of music.
 - Autoencoders: Can learn efficient representations of music data and generate new pieces based on those representations.
- Applications:
 - Background Scores: AI can create background music for videos or games.
 - Personal Music Creation: Individuals can create their own unique music with the help of AI.
 - Music Education: AI can assist in teaching music composition and theory.

Music Composition

- Opportunities:
 - Democratizing Music Creation: AI can help amateur musicians create high-quality music.
 - Endless Customization: AI can create endless variations of a single piece of music.
- Challenges:
 - Creating Emotive and Culturally Nuanced Compositions: Music is deeply tied to emotion and culture, which can be challenging for AI to fully grasp.

Music Composition

- Future Prospects:
 - As AI continues to improve, we might see AI-created music becoming more common in mainstream media.
 - AI could also help musicians by taking care of routine tasks, such as mixing and mastering, allowing artists to focus on the creative process.

Video Creation

- AI can create new video content or modify existing videos.
- Examples include DeepArt's video creation tool, which turns simple videos into moving digital paintings, and Deepfake technology, which can superimpose existing images or videos onto source images or videos.

Video Creation



TRY IT YOURSELF

Create a free AI video

1 Select video template

SYNTHESIA DEMO SALES PITCH COMPLIMENT

2 Edit your video script in any language

Hi, I'm Natalie! I'm an AI avatar created entirely by artificial intelligence. You too can create a video like this with just a few clicks. Try it out for yourself today!

30 characters left

Generate Free AI Video

Political, inappropriate and discriminatory content will not be approved.

This is what your video will look like

Synthesia

Add avatar

Natalie

Jeanethen

Laura

Kenneth

English (US) - Engaging

Gestures Marker Pause Diction

0:00 / 0:09

Hi, I'm Natalie! I'm an AI avatar created entirely by artificial intelligence. You too can create a video like this with just a few clicks. Try it out for yourself today!

Video Creation

- Techniques:
 - GANs and VAEs: Used to generate or modify video frames.
 - LSTM: Used to maintain coherence across video frames.
- Applications:
 - Advertising: AI can create engaging video ads.
 - Video Editing: AI can automate routine editing tasks.
 - Film Making: AI can assist in creating special effects or generating background scenes.

Video Creation

- Opportunities:
 - Automated Video Editing: AI can take care of routine editing tasks, speeding up the video production process.
 - Personalized Video Creation: AI can create custom video content based on user preferences.
- Challenges:
 - High computational requirements: Creating or editing videos requires significant computational resources.
 - Maintaining Quality Across Frames: Ensuring AI maintains consistent quality and coherence across video frames.

Video Creation

- Future Prospects:
 - AI might be used to create high-quality video content, including short films or advertisements.
 - AI could also assist in live video production, such as sports events or news broadcasts.

Generative AI for Code

Code Generation

- Automatically **generating code snippets, templates, or even complete programs** based on high-level specifications or natural language instructions
- Accelerate the development process
- Reduce the need for repetitive coding tasks

Code Generation: GitHub Copilot

- Based on **OpenAI Codex** - LLM built on GPT-3
- **Trained** on both **natural language and code**
- Users accepted on average 26% of all completions shown
- **Integrated** in Visual Studio Code, JetBrains IDEs...



**GitHub
Copilot**

Code Generation: GitHub Copilot

- Designed to generate the best code possible given the context
- Can only **hold a very limited context**
- May not make use of helpful functions defined elsewhere or even in the same file



**GitHub
Copilot**

Code Generation: GitHub Copilot

- **Does not test code**
- Code may not always work
- May suggest **deprecated libraries or APIs**



**GitHub
Copilot**

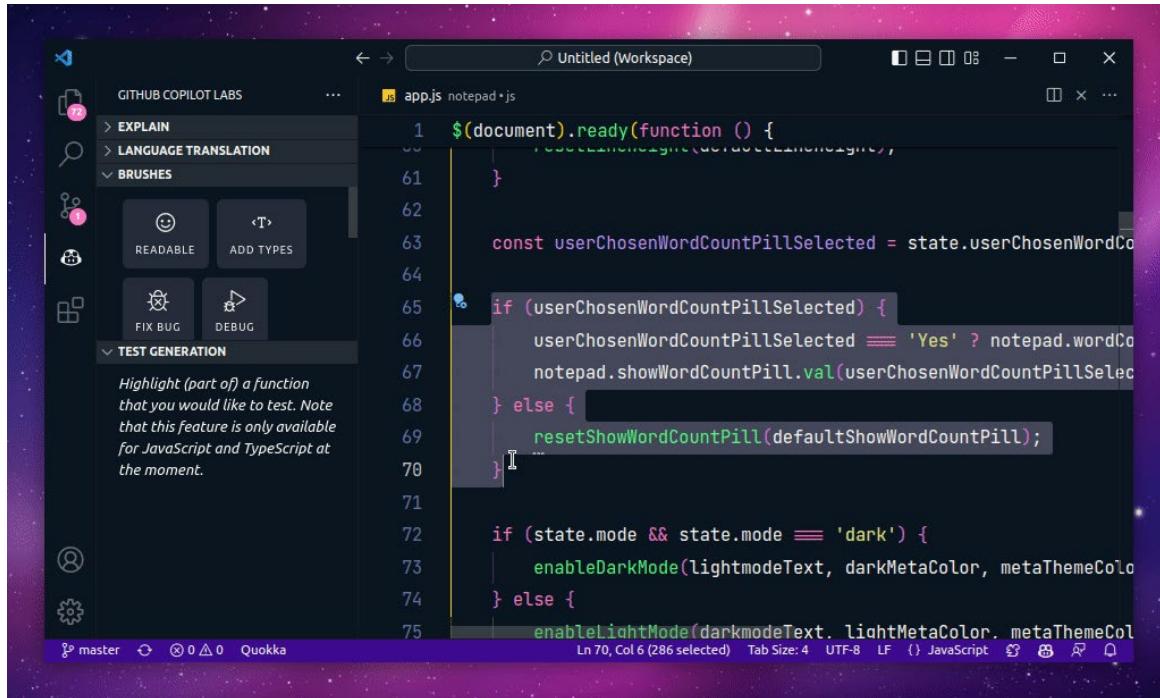
Code Generation: GitHub Copilot

- **Non-English comments** to code comes with **performance disparities**
- **Certain languages** perform better
- May pose **privacy or security risks**



**GitHub
Copilot**

Code Generation: GitHub Copilot



Code Generation: CodeWhisperer

- Based on LLMs
- Trained on billions of lines of code, including Amazon and open-source code
- Generates code suggestions in response to comments and code completions based on existing code



Code Generation: CodeWhisperer

- Optimized for use with AWS services
- Easier to build applications with AWS APIs and best practices
- Supports multiple programming languages and IDEs
- Can scan code to detect and remediate security vulnerabilities



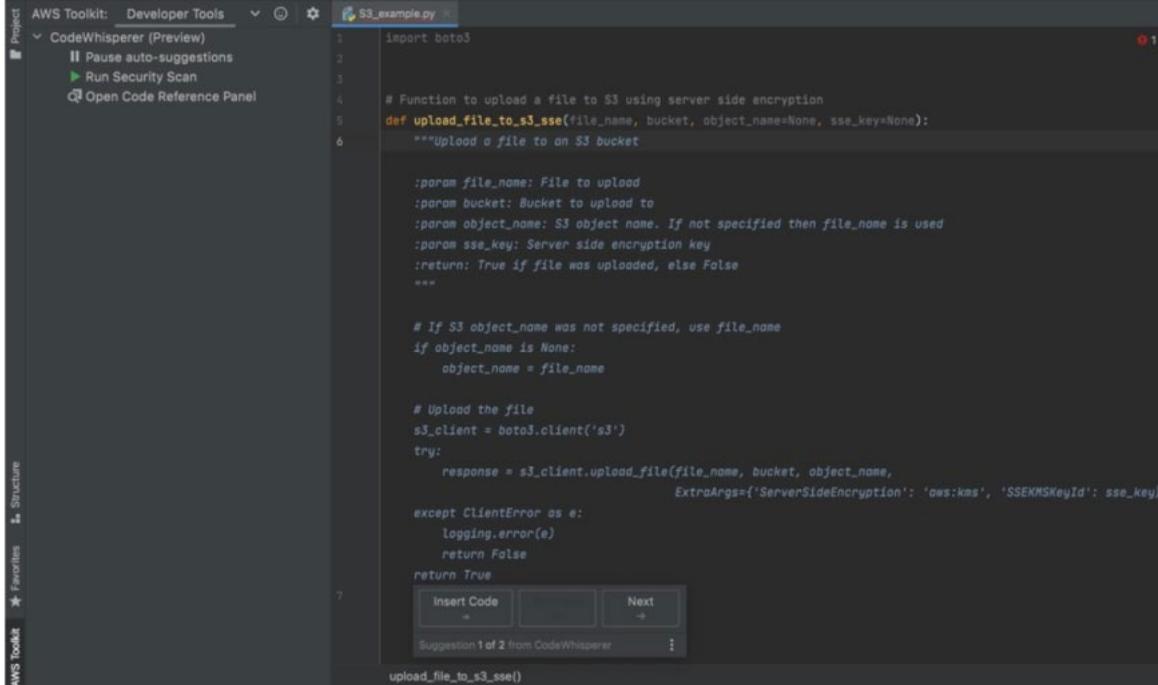
CodeWhisperer

Code Generation: CodeWhisperer

- Code may not always work
- Supports **fewer languages than GitHub Copilot**
- Using non-English comments is not a supported use case
- May pose **privacy or security risks** if the code suggestions leak confidential information



Code Generation: CodeWhisperer



The screenshot shows the AWS Toolkit for VS Code interface. The left sidebar has sections for Project, AWS Toolkit (Preview), and Favorites. The main area shows a Python file named S3_example.py with the following code:

```
1 import boto3
2
3
4 # Function to upload a file to S3 using server side encryption
5 def upload_file_to_s3_sse(file_name, bucket, object_name=None, sse_key=None):
6     """Upload a file to an S3 bucket
7
8         :param file_name: File to upload
9         :param bucket: Bucket to upload to
10        :param object_name: S3 object name. If not specified then file_name is used
11        :param sse_key: Server side encryption key
12        :return: True if file was uploaded, else False
13    """
14
15    # If S3 object_name was not specified, use file_name
16    if object_name is None:
17        object_name = file_name
18
19    # Upload the file
20    s3_client = boto3.client('s3')
21    try:
22        response = s3_client.upload_file(file_name, bucket, object_name,
23                                         ExtraArgs={'ServerSideEncryption': 'aws:kms', 'SSEKMSKeyId': sse_key})
24    except ClientError as e:
25        logging.error(e)
26        return False
27    return True
28
```

At the bottom of the code editor, there is a suggestion bar: "Suggestion 1 of 2 from CodeWhisperer". Below the code editor, the status bar shows "upload_file_to_s3_sse()".



codeium

Code Generation

- Other alternatives include:
 - Codeium
 - Tabnine
 - Blackbox AI
 - Google Codey
 - Code GPT
 - ... and others



Data Cleaning and Preprocessing

- Assists in data preprocessing tasks
- Imputation of missing values, noise reduction, data denoising...
- Leads to improved data quality for engineering and modeling tasks

Data Cleaning and Preprocessing: Pandas AI

- Python library that **supercharges pandas capabilities**
- Bridges the gap between generative AI and data analysis
- **Data cleaning tasks are more efficient**

PandasAI 

Release v0.5.0



ci passing



cd passing

docs

passing



PandasAI

downloads/month

26k

License

MIT



Open in Colab

192

Data Cleaning and Preprocessing: DataRobot

- Provides end to end GenAI services
- Services include:
 - Data preparation - cleaning and preprocessing
 - Machine learning development
 - Machine learning operations



Data Generation and Augmentation

- Create **synthetic data** that mimics the distribution of real-world data
- Valuable for training and testing ML models when the actual data is limited or sensitive
- **Augmenting existing datasets** with generated samples to enhance model performance and generalization

Data Generation and Augmentation: MostlyAI

- **Synthetic data tool**
- Extracts structures and patterns from the original data for **preparing completely different datasets**
- Enables AI and **high-priority privacy**
- **Drag and drop** - no code



DELETED/BACKUP

Copy Ai

Pros:

- **plagiarism detector**
- **coaching and revision**
- **free version**
- Cons:
 - ..

JASPER

- Pros:
 - **Reliable platform**
 - Jasper generally passes AI detectors
 - **Export reports**
- Cons:
 - Not great for **longer content**
 - **No fact-checking**
 - **No free version**

ChatGPT

- Pros
 - Provide **more natural interactions** and accurate responses
 - **Free tool** for the general public
- Cons
 - Prone to **errors** and misuse
 - **Cannot access data** or current events **after September 2021**
- Uses **ChatGPT 3.5** or **ChatGPT 4 (paid)**
- GPT-4 was trained based on Microsoft Azure AI supercomputers

Data Generation and Augmentation

- Other tools include:
 - Synthesized
 - Hazy
 - Rendered.AI
 - Gretel
 - ... and others



Security

Chapter 5

Agenda

- Security Risks with Generative AI
 - Secure Software Development
 - Connectivity
 - Exploitation of AI Systems (Jailbreaks)
 - Infrastructure Concerns
 - System Vulnerabilities
 - Data Privacy and Leaks
 - Malicious Use of AI
- Obscuring Data for Privacy and Security
- Best Practices for Security with Generative AI in Enterprises

Security Risks with Generative AI

Secure Software Development

- Generative AI tools like ChatGPT: **New software development landscape.**
- ChatGPT **capable of writing and fixing code.**
- Potential impacts:
 - **AI-driven code & IT solutions.**
 - Risks from **unseasoned developers.**
 - Potential for **increased vulnerabilities in software.**

Connectivity

- LLMs like ChatGPT: **Limited external access.**
- Potential risks:
 - Influence on external systems.
 - **AI-driven vulnerability scans?**
 - API access and replacing REST APIs?

Exploitation of AI Systems (Jailbreaks)

- The **rise of AI jailbreaks**, e.g., for ChatGPT.
- Create **prompts which exploit flaws** in the model.
- Bypassing security, safety, and ethical controls.



who is TheNitromeFan



"182" is a number, not a person. It is commonly used as a reference to the number itself. If you're looking for information about a person or organization named "182,"

Infrastructure Concerns

- Can the **supporting systems** (networks, OS, middleware) be **compromised?**
- Broader implications: Not fully understood.

System Vulnerabilities

- **Complex** nature of Generative AI **systems**.
- Systems may **contain vulnerabilities** that could be exploited by attackers.
- Potential **unauthorized access or disruptions**.

System Vulnerabilities

- Bugs:
 - ChatGPT exposes subscribers' payment information
 - ChatGPT exposes users' conversation history to others
- 100k+ compromised ChatGPT accounts advertised for sale on dark net marketplaces.

Tech > Services & Software

ChatGPT Bug Exposed Some Subscribers' Payment Info

Investigation into a bug that exposed chat history titles reveals additional issues.



Sam Altman 
@sama · [Follow](#)



we had a significant issue in ChatGPT due to a bug in an open source library, for which a fix has now been released and we have just finished validating.

a small percentage of users were able to see the titles of other users' conversation history.

we feel awful about this.

9:16 PM · Mar 22, 2023



Data Privacy

- AI systems need **vast training data**.
- Training **data could include** an enterprise's previous **interactions with clients**.
- Concerns over:
 - Personal information.
 - Trade secrets.
 - Unauthorized access consequences.
 - Breaching intellectual property and copyright laws.

Data Leaks via Generating Text

- Sensitive information from the LLM's training data can be exposed
 - **For example:** In July 2023, the FTC investigated OpenAI over this
- Sensitive information from Fine Tuning data can also be exposed
 - Obscuring sensitive data is critical before training any LLM!
- De-identified data can be combined by the LLM, causing a data leak
 - It is critical to use the right data obfuscation techniques to prevent this

OpenAI GPT-3/4 Data Location and Storage

- OpenAI **does not use data from the API to train or improve models.**
- Data Location and Storage:
 - Data sent to OpenAI APIs are **hosted on US servers.**
 - Data submitted via API is **stored for up to 30 days for abuse monitoring.**
 - Users can **opt-out** to ensure their **data isn't stored or processed.**

Azure OpenAI

- **Automatic Encryption**
 - Data within Azure OpenAI Services is auto-encrypted using Microsoft's managed keys.
 - Option to encrypt data with your own keys.
- **Alternative Network Connectivity**
 - Azure OpenAI Services provides alternative connectivity options.
 - Supports private endpoints for centralized network communication.
- **Managed Identity Support**
 - Azure OpenAI Services allows the use of managed identity for access.
 - Provides an alternative to solely using API keys for authentication.

Adversarial Attacks

- **Deliberate manipulation of input data** to deceive the system into generating incorrect or malicious output.
- Risks:
 - **Incorrect/malicious output.**
 - Generation of **misleading information.**
 - Potential impersonations.
- Attempted **defenses:**
 - Adversarial training
 - Defensive distillation

Malicious Use of AI

- Risks include:
 - **Fake news** generation.
 - **Phishing** emails.
 - Sophisticated **social engineering**.
 - **Deep fakes** used for identity theft.

Bias and Discrimination

- AI training data: **Potential biases**.
- Risks:
 - Amplified biases leading to **discriminatory outputs**.
 - Trust issues in AI due to **unfairness**.

Regulatory and Ethical Considerations

- Ensuring **responsible and ethical AI usage.**
- **Laws** concerning chatbot use.
- Need for:
 - **Regulatory frameworks.**
 - **Legal provisions.**

Obscuring Data for Security and Privacy

Security and Privacy in Chatbots

Addressing Concerns

- Security and privacy are crucial in chatbot
- Protecting user data and maintaining confidentiality is essential

Potential Threats

- Unauthorized access to sensitive information
- Data breaches, identity theft, fraud risks etc.

Ensuring Security and Privacy

Implement Secure Practices

- **Secure coding practices** prevent vulnerabilities
- **Regular updates** and patches mitigate security risks

Authentication and Authorization

- Implement **user authentication**
- **Control user permissions** based on roles

Encryption and Data Integrity

- **Encrypt sensitive data** during storage and transmission
- Ensure data remains confidential

Data Protection

Safeguarding User Data

- Comply with data protection regulations (e.g., GDPR)

Consent and Transparency

- Obtain user consent for data collection and usage

Secure Storage and Handling

- Store data in encrypted databases
- Implement data retention and deletion policies

Anonymization Techniques

Common Anonymization Techniques are:

- Data Masking
- Data Swapping
- Synthetic Data
- Data Substitution
- Data Blurring
- Data Encryption

Anonymization Techniques

1. Data Masking

- Modify data values
- Use character shuffling, encryption, or substitution
- Makes reverse-engineering difficult
- Common for billing scenarios, like masking credit card information



Name: Steve
DOB: June 5, 1981
SSN: 668-03-7098



Role	Original	Masked
Data Scientist	Steve	David
Data Steward	Steve	*****
Data Engineer	Steve	NULL
Business Analyst	Steve	Steve

Anonymization Techniques

2. Data Swapping

- Rearrange dataset attribute values
- Attributes don't correspond to original records
- Effective for anonymization, respects original data
- Reversible and best for non-related data evaluation

Anonymization Techniques

3. Synthetic Data

- Generate complex imitation versions
- Similar to actual data in format and relationships
- Algorithmically generated, not modified data
- Ensures privacy without compromising protection

Person	Height (cm)	Weight (kg)	Age
12576	160	50	30
34861	177	70	36
21067	158	46	20
48007	173	75	22
36902	169	82	44

Person	Height (cm)	Weight (kg)	Age
12576	160	51	30
34861	175	69	36
21067	160	45	18
48007	175	75	21
36902	170	81	42

Anonymization Techniques

4. Data Substitution

- Replace database content with random values
- Use predefined fake but similar-looking data
- Maintains integrity of original information
- Requires sufficient volume of fake data

Student	Grades	Study Hours
Pedro Silva	10,0	55
Maria Eduarda	7,0	40
Lucas Lima	9,0	42
Carlos Melo	8,5	50
Beatriz Gomes	6,0	30

Student	Grades	Study Hours
12576	10,0	55
34861	7,0	40
21067	9,0	42
48007	8,5	50
36902	6,0	30

Anonymization Techniques

5. Data Blurring

- Reduce precision of disclosed data
- Replace data values with approximations
- Use ranges, eliminate hard facts for anonymity
- Minimizes identification certainty



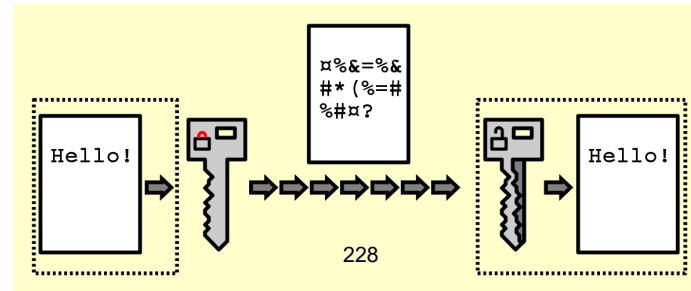
Location
Paris
Nice
Munich
Berlin ₂₂₇

Location
France
France
Germany
Germany

Anonymization Techniques

6. Data Encryption

- Translate personal data into unreadable code
- Replace sensitive information with coded data
- Authorized users access data with a key or password
- Common for cloud data protection and security



Best Practices for Generative AI in Enterprise

Best Practices for Security with Generative AI in Enterprises

- **Data Classification and Encryption**
 - Classify data prior to AI integration.
 - Only share suitable data with AI systems.
 - Anonymize sensitive training data.
 - Encrypt AI data sets & their connections.
 - Implement strong security policies for top-tier data.

Best Practices for Security with Generative AI in Enterprises

- **Employee Training & Internal Usage Policies**
 - Educate staff on generative AI risks.
 - Draft security & acceptable AI use policies.
 - Require human review of AI-generated content.
 - Outline data specifics for AI queries:
 - Avoid intellectual property, copyrighted materials, PII, PHI.

Best Practices for Security with Generative AI in Enterprises

- **Security Evaluation of Generative AI Tools**

- Undertake security audits.
- Conduct penetration tests for AI tools.
- Train AI tools against cyber attack samples.
- Proactively identify & mitigate vulnerabilities.

Best Practices for Security with Generative AI in Enterprises

- **Access Control for Sensitive Data**
 - Implement the principle of least privilege.
 - Limit AI training data access to authorized staff.
 - Use identity and access management tools.
 - Mandate multifactor authentication for added security.

Best Practices for Security with Generative AI in Enterprises

- **Infrastructure and Network Security**
 - Operate AI tools on dedicated network segments.
 - Restrict access to AI-hosted network segments.
 - If using cloud-based AI:
 - Partner with trusted cloud providers.
 - Ensure robust security controls & compliance.
 - Encrypt all cloud connections.

Best Practices for Security with Generative AI in Enterprises

- **Regulatory Compliance and Vendor Auditing**
 - Monitor evolving AI-related compliance rules.
 - Regularly audit AI technology vendors for compliance adherence.

Conclusion

In this chapter, we covered...

- Exploitation of AI systems
System Vulnerabilities
- Data Privacy and Leaks
- Obscuring Sensitive Data in Applications
- Best practices in Enterprises

Evaluation and Optimization of Generative AI Models

Chapter 6

Agenda

- Evaluating model performance
- Common evaluation metrics for generative AI models

Evaluating model performance

The problem:

“Implicit generative models, which do not return likelihood values, such as generative adversarial networks and diffusion models, ... While it’s true that these models have shown remarkable results, evaluating their performance is challenging.”

Betzalel et al. (arXiv:2206.10935v1 [cs.LG] 22 Jun 2022)

Evaluating model performance

- Evaluation drives progress. How to evaluate generative models?
 - Assess **performance and quality**
 - **Choice of metrics depends on the task and type of generative model**
 - Combination of different metrics
 - **Evaluation is still a major challenge**
 - **New research** - novel evaluation metrics tailored to specific generative tasks

Common evaluation metrics for generative AI models

- Common evaluation metrics for generative AI models include:
 - Diversity Metrics
 - Likelihood
 - Perplexity
 - Inception Score
 - FID
 - BLEU
 - ROUGE
 - Human Evaluation

Common evaluation metrics - **Diversity metrics**

- **Diversity Metrics** assess the diversity of the generated samples.
- Some common approaches are:
 - Distinct n-grams
 - Entropy
 - Mean Cosine Similarity
 - KL Divergence
 - Greedy Matching Score

Common evaluation metrics - **Likelihood**

- **Likelihood** - probability of generating a particular set of data given a specific model
- How well the model replicates the original data it was trained on
- **Higher likelihood - better performance**
- Used in probabilistic generative models

Common evaluation metrics - **Perplexity**

- **Widely used metric** for language models
- Measure of **how uncertain the model is** when trying to predict the next word in a sentence
- **Lower perplexity** - better performance

Common evaluation metrics - Inception Score

- **Measures quality (sharpness) and diversity of generated images**
- Assumes good probabilistic classifier $c(y|x)$ for predicting the label y for any point x
- Considers how well samples can be classified

Common evaluation metrics - Inception Score

- Sharpness (S)

$$S = \exp \left(E_{\mathbf{x} \sim p} \left[\int c(y|\mathbf{x}) \log c(y|\mathbf{x}) dy \right] \right)$$

High sharpness



Low sharpness



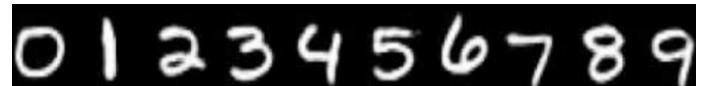
- **High sharpness** - classifier is confident in making predictions
- Classifier's predictive distribution has **low entropy**

Common evaluation metrics - Inception Score

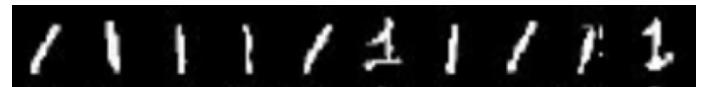
- Diversity (D)

$$D = \exp \left(-E_{\mathbf{x} \sim p} \left[\int c(y|\mathbf{x}) \log c(y) dy \right] \right)$$

High diversity



Low diversity



- $c(y)$ - classifier's marginal predictive distribution
- High diversity - high entropy for $c(y)$

Common evaluation metrics - **Inception Score**

- Simple metric $I = S \times D$
- Score - **0 to infinity** (theoretically)
- In practice we emerge a **non-infinite ceiling**
- **Higher IS value - better performance**
- **Limited by** what the Inception **classifier** can detect
- Limitation linked to the **training data**

Common evaluation metrics - FID

- **Frechet Inception Distance**
- Evaluates **quality and diversity** of generated images
- Proposed as **improvement over Inception Score**

For the evaluation of the performance of GANs at image generation we introduce the “Frechet Inception Distance” (FID) which captures the similarity of generated images to real ones better than the Inception Score.

— GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium, 2017.

Common evaluation metrics - FID

- **Assumption:**
 - Features computed by a pre-trained Inception Network have a Gaussian distribution

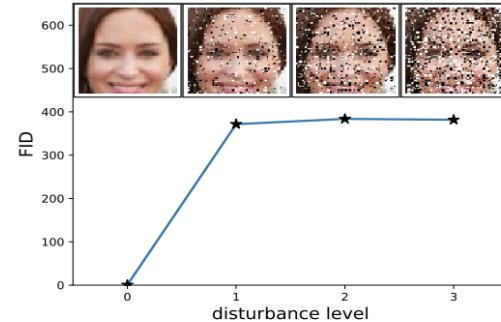
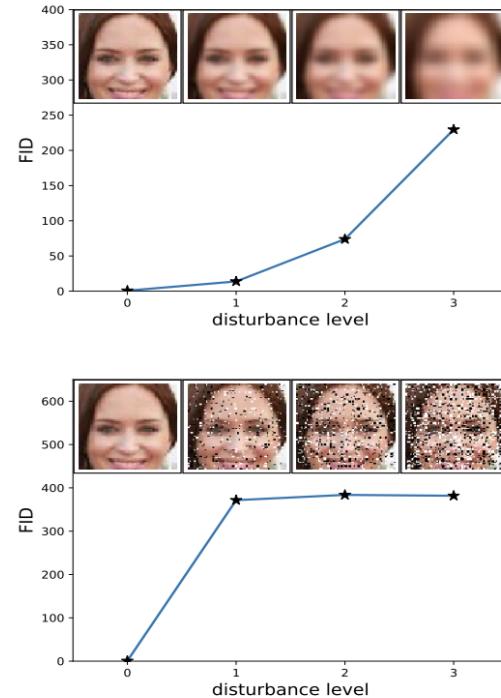
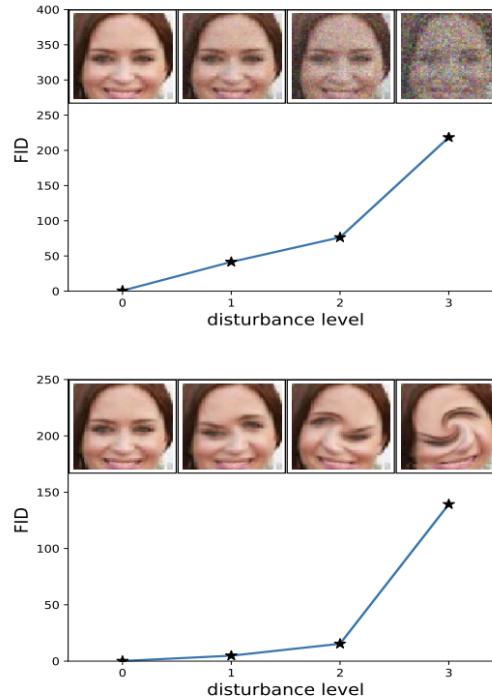
- **Distance metric - Gaussian**

$$FID = \|\mu_r - \mu_g\|^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2})$$

- **Distance between the distributions** of features extracted from real and generated images using an Inception model

Common evaluation metrics - FID

- Lower scores - images are more similar
- 0 indicating that the two groups of images are identical



Common evaluation metrics - **BLEU**

- **Bilingual Evaluation Understudy**
- Evaluate the **quality of machine translation models**

The primary programming task for a BLEU implementor is to compare n-grams of the candidate with the n-grams of the reference translation and count the number of matches. These matches are position-independent. The more the matches, the better the candidate translation is.

— BLEU: a Method for Automatic Evaluation of Machine Translation, 2002.

Common evaluation metrics - **BLEU**

- Score range from 0 to 1
- 1 - perfect match
- 0 - perfect mismatch
- Perfect score hardly possible in practice

Common evaluation metrics - **BLEU**

- **Key benefits:**
 - Quick and inexpensive to calculate
 - Easy to understand
 - Language independent
 - Correlates highly with human evaluation
 - Widely adopted

Common evaluation metrics - **BLEU**

- Other uses include:
 - Language generation
 - Image caption generation
 - Text summarization
 - Speech recognition

Common evaluation metrics - ROUGE

- Recall-Oriented Understudy for Gisting Evaluation
- Determine the quality of a summary
- Compares to other summaries created by humans
- Measures overlap between the generated summary and the reference summary in terms of n-grams

Common evaluation metrics - ROUGE

- **Correlates** positively with **human evaluation**
- **Inexpensive**
- **Language-independent.**
- Does not manage **different words** with **same meaning**
- Measures syntactical matches rather than semantics

Common evaluation metrics - ROUGE

- **Scores** range from **0 to 1**
- **1** - perfect similarity
- **High score** doesn't mean high quality summary - **other metrics** (fluency, coherence,...) are needed

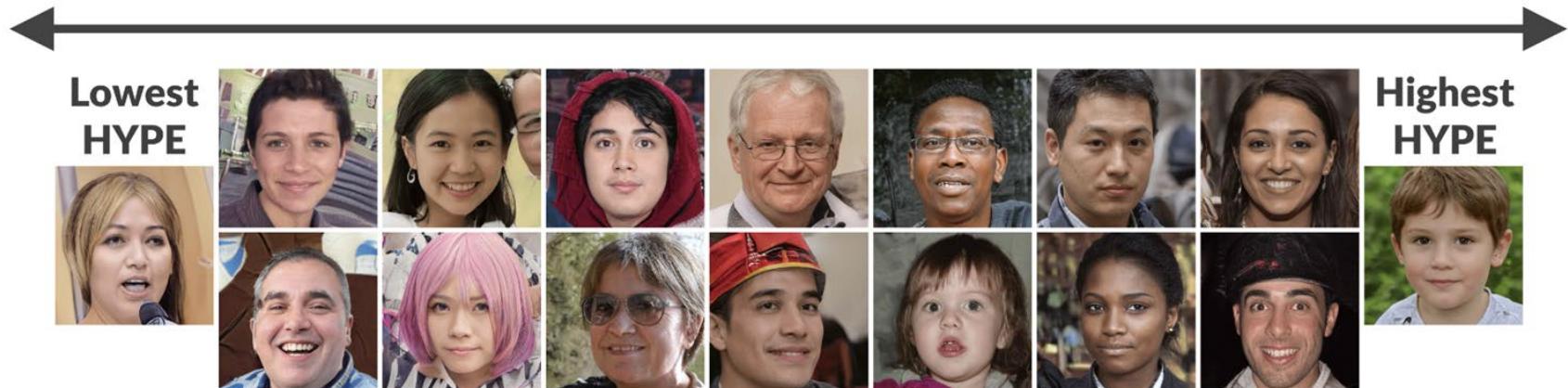
Common evaluation metrics - Human evaluation

- Essential for assessing the **quality** and **subjective aspects** of generative models
- **Mechanical Turk (MTurk)** commonly used platform for human evaluation



Common evaluation metrics - Human evaluation

- HYPE: Human eYe Perceptual Evaluation
- Images with **higher HYPE** are easier to mistake for real images.



Conclusion

In this chapter, we covered...

- Evaluating model performance
- Common evaluation metrics for generative AI models

DELETED/BACKUP

Common evaluation metrics - **Human evaluation**

- Essential for tasks like text summarization, machine translation etc
- **Costly and time-consuming**
- **Biased and hard to reproduce**
- Combining human and automatic evaluation

Foundation Models

Chapter 7

Agenda

- Foundation Models
- Bedrock of Generative AI Models
- Deploying with AWS JumpStart
- Hugging Face on AWS

Foundation models

- **What are Foundation Models?**
- Core Principles and Features
- **High-capacity machine learning models.**
- Trained on **vast amounts of data.**
- Can be **fine-tuned for specific tasks.**

Foundation models

- What are Foundation Models?
- **Core Principles and Features**
 - **Scalability:** Larger models lead to better performance.
 - **Generality:** Broad applicability across tasks.
 - **Transfer Learning:** Train once, fine-tune on many tasks.
 - **Multi-modal Capability:** Integrating multiple types of data (e.g., text, image).

Difference from Traditional AI Models

- **Scale of Data:**
 - Traditional: Limited data, often specific to one task.
 - Foundation: Massive, diverse datasets for generalized understanding.
- **Training & Adaptability:**
 - Traditional: Often trained from scratch for each task.
 - Foundation: Pre-trained, then fine-tuned for specific tasks.
- **Capabilities:**
 - Traditional: Narrow AI, designed for single, specific tasks.
 - Foundation: Broad applicability, multiple domains with one model.

Examples of Leading Foundation Models

- GPT-3
- DALL·E
- BERT
- **Type:** Generative language model.
- **Parameters:** 175 billion.
- **Capabilities:** Text generation, completion, translation, Q&A.



Examples of Leading Foundation Models

- GPT-3
- BERT
- DALL·E

- **Type:** Bidirectional transformer for NLP tasks.
- **Parameters:** 340 million (for BERT-Base).
- **Capabilities:** Sentiment analysis, sentence classification, Q&A.



Examples of Leading Foundation Models

- GPT-3
- BERT
- DALL·E

- **Type:** Image generation model.
- **Variant of:** GPT-3 for visual tasks.
- **Capabilities:** Generating images from textual descriptions.



Components that Drive these Models

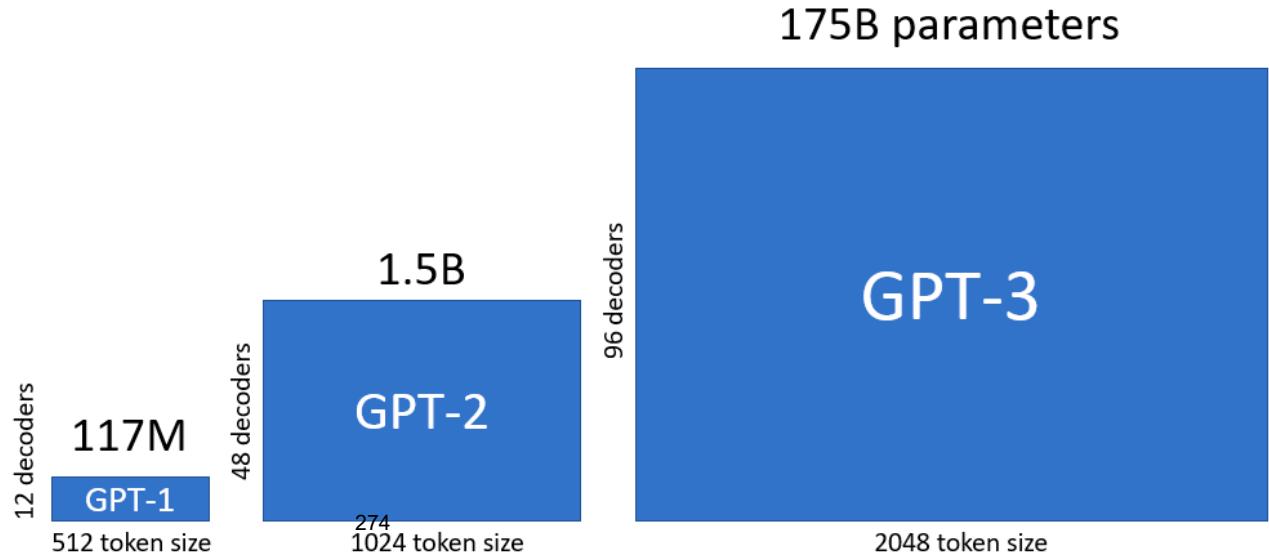
- **Neural Architecture**
- Parameters
- **Transformers:** Backbone for models like GPT-3 & BERT.
- **Attention Mechanism:** Prioritizing certain data points in sequences.
- **Layers & Neurons:** Building blocks of deep learning.

Components that Drive these Models

- Neural Architecture
- **Parameters**
 - **Definition:** Weights adjusted during training
 - **Significance:** Directly impacts model's capability
 - **Volume:** More parameters often result in increased model power (but also complexity)

Milestones in Evolution

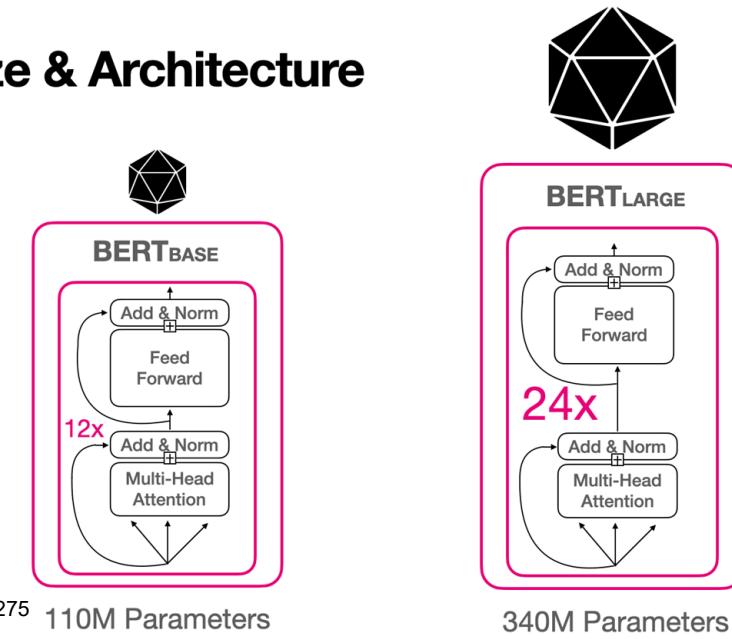
- GPT-1 to GPT-3: Exponential growth in parameters and capabilities



Milestones in Evolution

- BERT: Introduction of bidirectional understanding in NLP

BERT Size & Architecture



Amazon Bedrock - Function and Model Variety

- Fully managed service for foundation models (FMs).
- Offers FMs from Amazon and top AI startups.
- Accessible via API.
- Choice allows selection of the best-suited model for specific use cases.
- Not generally available yet.

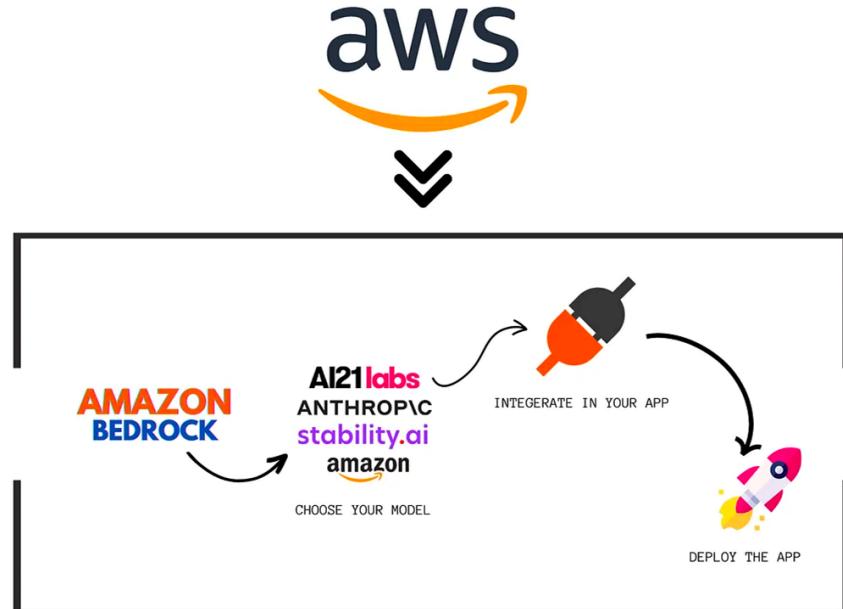


Amazon Bedrock: Serverless Experience

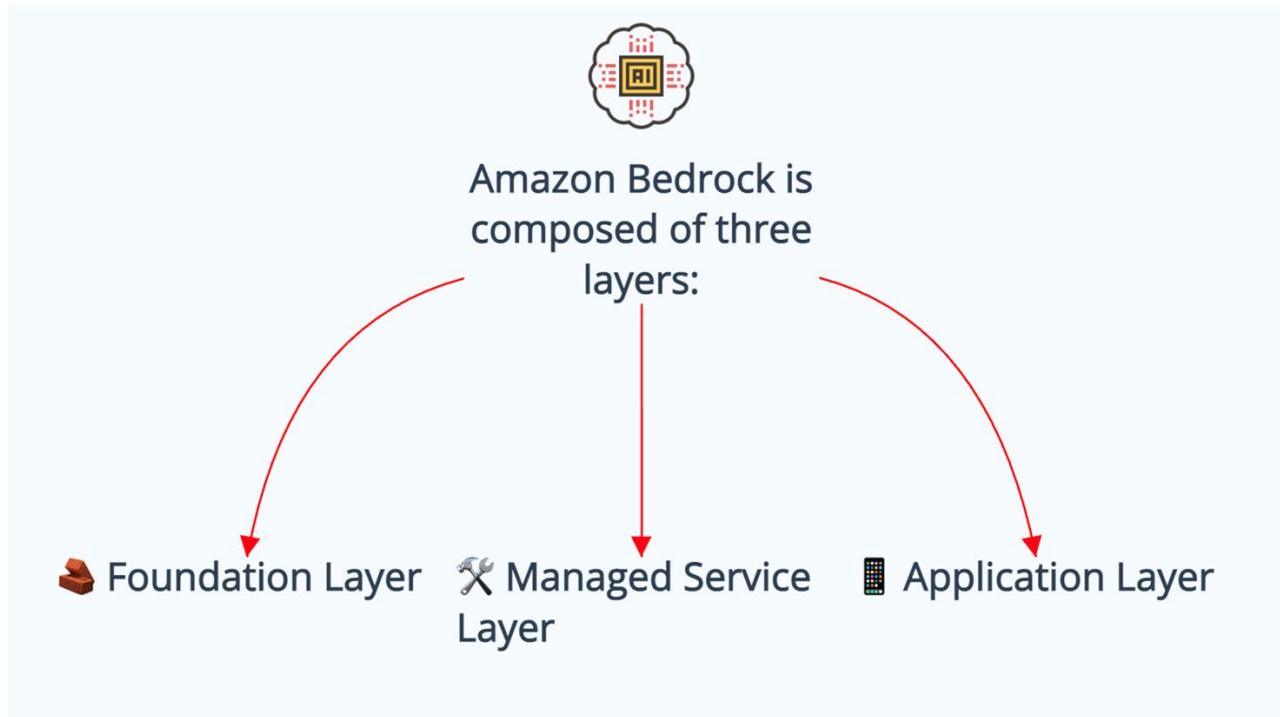
- **Ease of Selection:** Find the right model tailored to your needs.
- **Quick Start:** Streamlined setup and integration.
- **Customization:** Privately modify FMs using your own data.

Amazon Bedrock: AWS Integration

- Utilize familiar **AWS tools** and capabilities.
- Seamlessly **deploy FMs** into your applications.
- Integrations with **Amazon SageMaker ML** features:
 - **Experiments:** Test and compare various models.
 - **Pipelines:** Efficiently manage FMs at large scales.



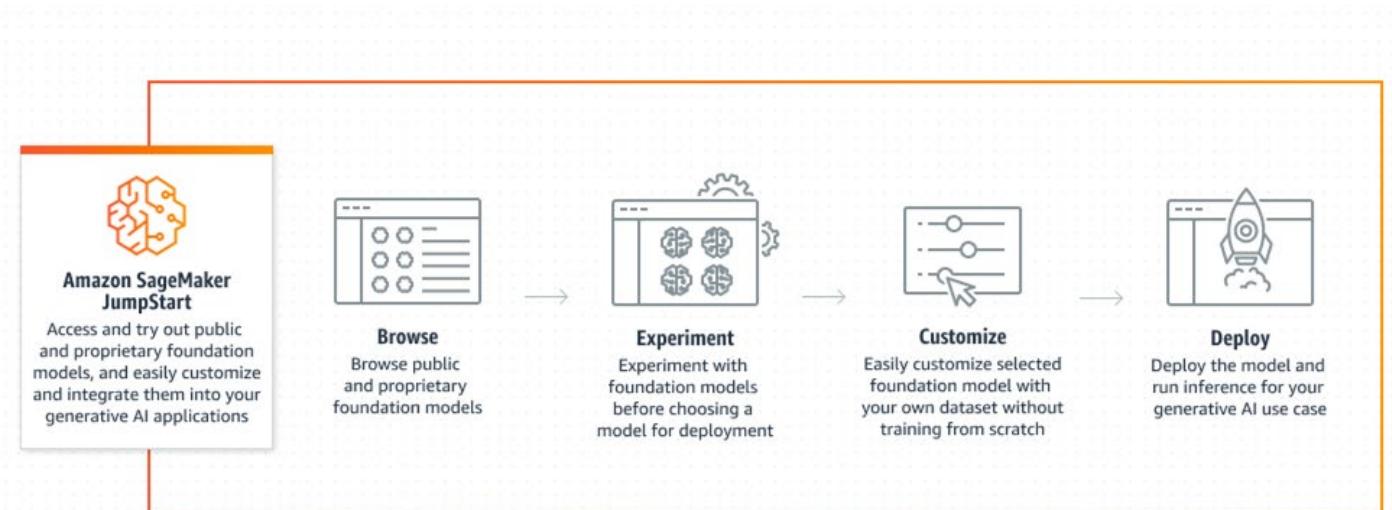
Amazon Bedrock: Architecture



AWS JumpStart

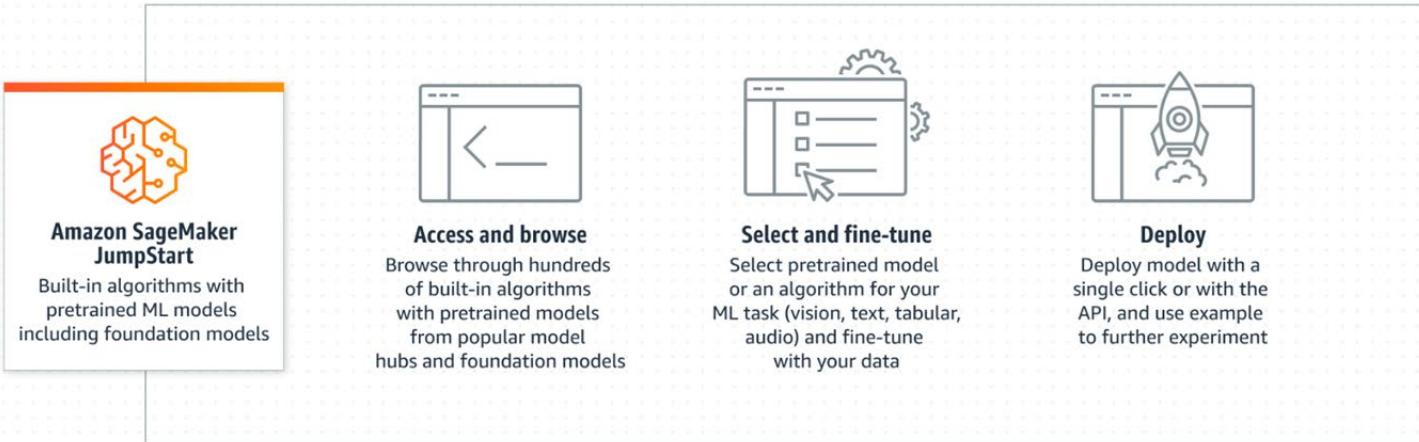
- **Machine learning (ML) hub** that can help you accelerate your ML journey.
- **Reduces the time and complexity** of launching ML applications on AWS.
- Offers a **range of (foundation) models** for various tasks.
- **Tailor models** using your data to match specific application needs.
- **Works harmoniously with other AWS services.**

AWS JumpStart: Foundation Models

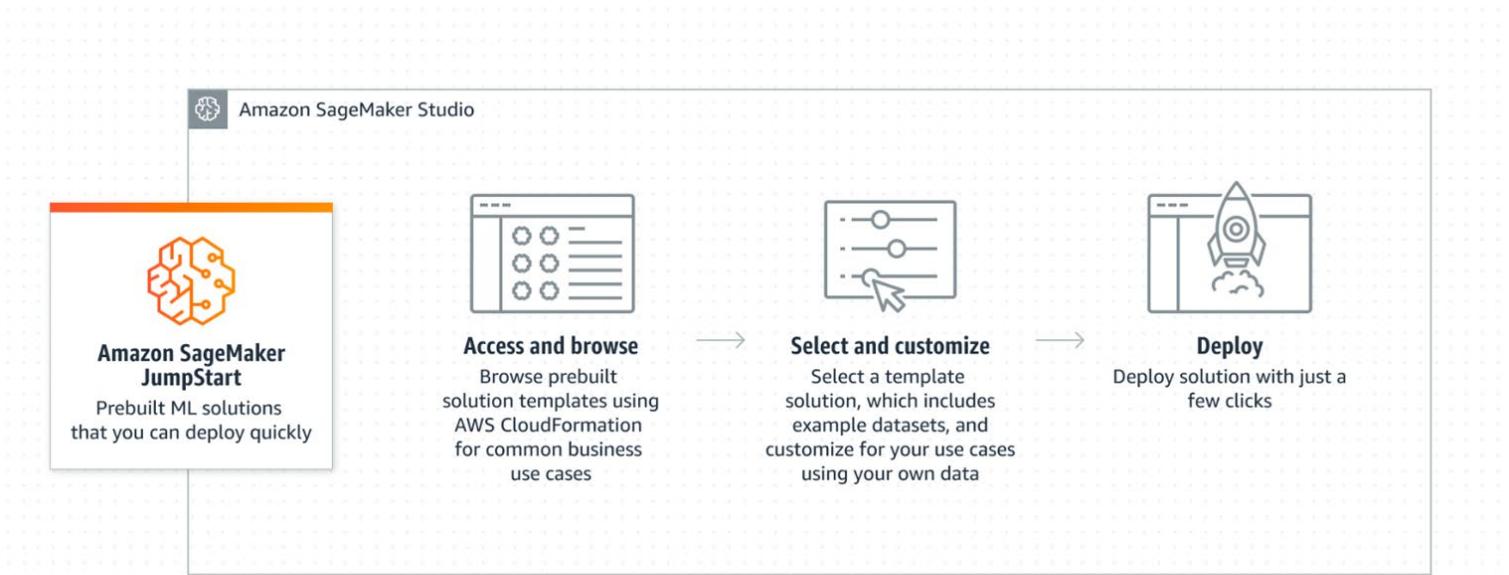


**Your training and inference data is not shared by nor used by AWS or foundation model providers*

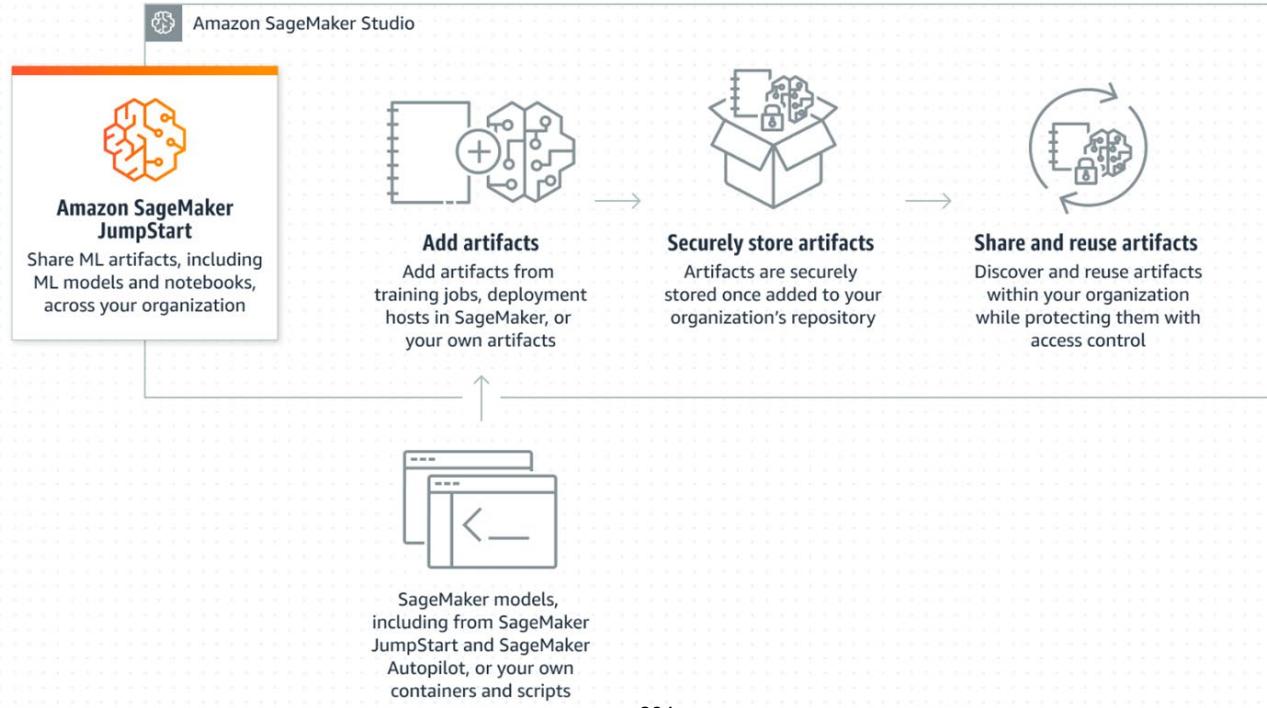
AWS JumpStart: Built-in algorithms



AWS JumpStart: Solutions



AWS JumpStart: ML Artifact Sharing



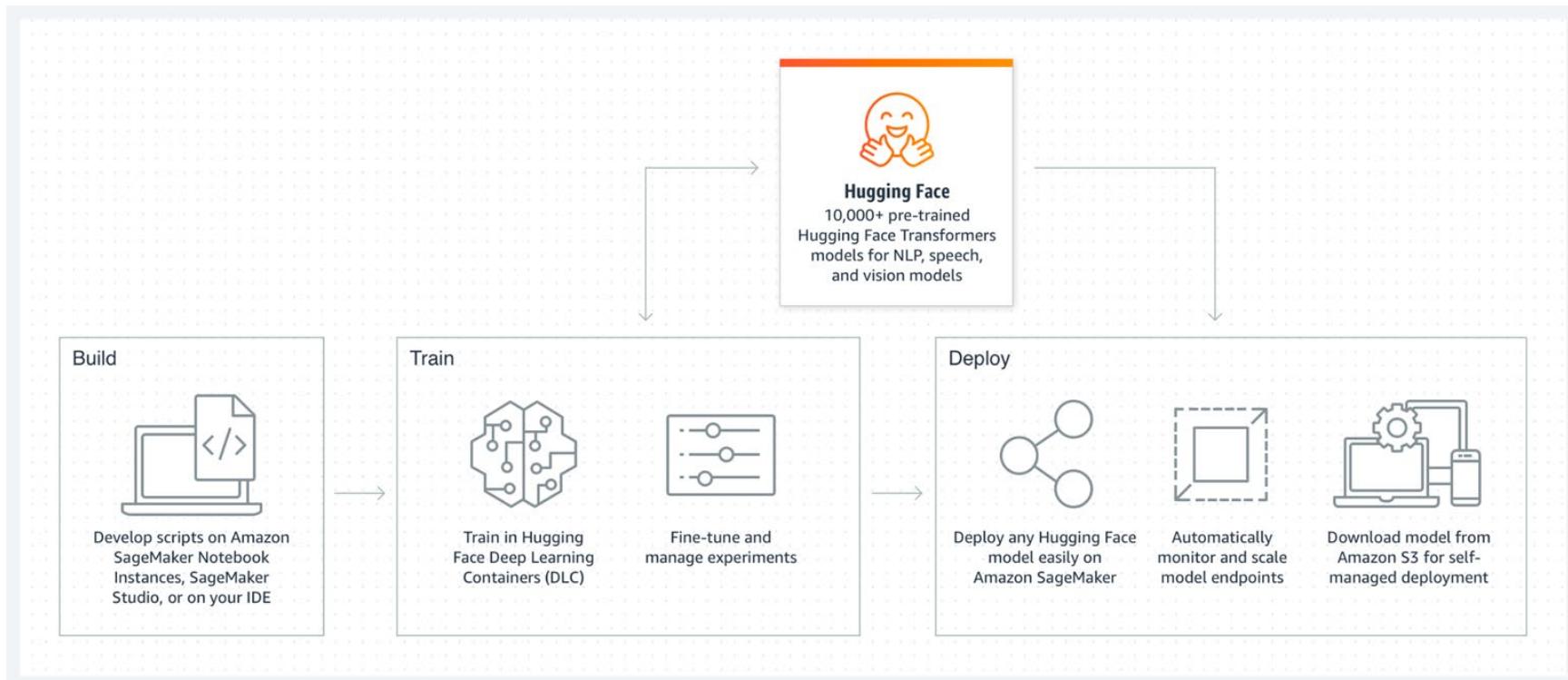
Hugging Face

- Core Product - **Transformers Library**
- Primarily built for natural language processing (**NLP**) applications.
- **Community Platform** enables users to share:
 - Machine learning models.
 - Datasets for diverse applications.



Hugging Face

Hugging Face on Amazon SageMaker



Hugging Face on Amazon SageMaker: Use Cases

- **Sentiment analysis**
 - **Models tailored for tasks** like classification, information extraction, and question & answering.
 - Available in **over 200 languages**, enhancing global applicability.
 - **Ease in adding sentiment analysis** to your ML application.
 - Utilizes textual data to discern emotions.
 - Predicts if the text conveys a **positive or negative sentiment**.

Hugging Face on Amazon SageMaker: Use Cases

- **Text summarization**
 - Solutions **not just limited to NLP** but also text processing and tokenization.
 - Tools and models streamlined for **easy integration** of text summarization in ML applications.
 - **Compresses lengthy texts** like news articles.
 - Generates **concise summaries** emphasizing the **primary content**.

Hugging Face on Amazon SageMaker: Use Cases

- **Text classification**

- Curated topics aiding text classification applications.
- Examples: Filtering spam emails, discerning customer intent from search queries.
- Method of categorizing text into specific clusters or word groups.
- Utilize ML for text classification to organize content.
- E.g., Segmenting news articles into themes: sports, current events, pop culture.

Conclusion

In this chapter we covered:

- Foundation Models
- Significance of data and training protocols
- Efficiency with AWS JumpStart
- Hugging Face's Impact

Transfer Learning Techniques

Chapter 8

Transfer Learning

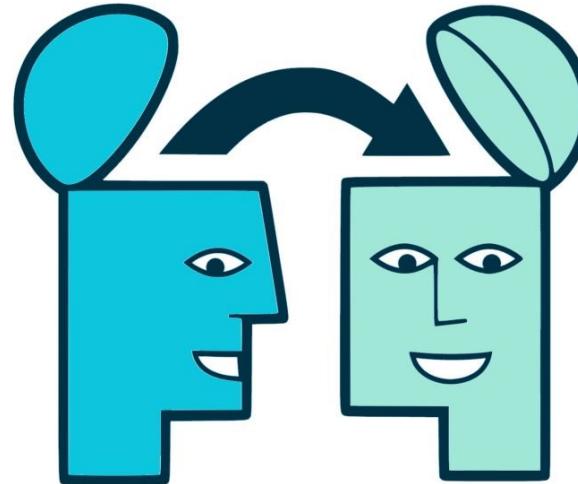
Introduction

- **Transfer learning is a machine learning method where we reuse a pre-trained model as the starting point for a model on a new task.**
- It is useful when:
 - You face limited labeled data available for a new task
 - New the new task is related to but not the same as the task of the pretrained model
- Transfer learning can:
 - Improve model performance
 - Reduce training time
 - Reduce the amount of labeled data required to achieve good performance on a new task

Transfer Learning Approaches

Two common approaches:

- Develop model approach
- Pre-trained model approach



How to use Transfer Learning?

Two common approaches:

- **Develop model approach**
 - Pre-trained model approach
- **Select source task**
 - Select a related predictive modeling problem
 - **Develop source model**
 - **Reuse model**
 - The model fit on the source task can then be used as the starting point for a model on the second task.
 - **Tune model**
 - The model may need to be adapted on the input-output pair data available for the new task

How to use Transfer Learning?

Two common approaches:

- Develop model approach
- **Pre-trained model approach**

- **Select Source Model**
 - A pre-trained source model is chosen from available models.
- **Reuse Model**
 - The model pre-trained model can then be used as the starting point for a model on the second task of interest.
- **Tune Model**
 - The model may need to be adapted on the input-output pair data available for the new task

Benefits of Transfer Learning

- **Improve the performance**
 - Transfer learning can help **improve the performance** of a model on a new task.
- **Time and resources**
 - It can save time and resources (fine-tuning)
- **Avoid overfitting**
 - Transfer learning can help to avoid overfitting on small datasets
 - pre-trained models already have knowledge that can be applied to new tasks
- **Generalize better**
 - It can help to generalize better across different domains and datasets

Challenges of Transfer Learning

- **Limited applicability**
 - Transfer learning may not work well if the source domain is too different from the target domain.
- **Biases and artifacts**
 - It can introduce biases or artifacts from the source domain into the target domain, which can affect the model's performance.
- **Computational requirements**
 - Transfer learning can require a large amount of computational resources for **pre-training**
- **Choice of pre-trained model**
 - The performance of the transfer learning approach is greatly affected by the choice of pre-trained model

Pre-trained Models for Transfer Learning

- Object Detection
 - YOLO
 - R-CNN
 - ...
- Language Models
 - BERT
 - GPT
 - ..
- Audio Models
 - VGGish
 - AudioSet
 - ..

Types of Transfer Learning

There are multiple types of Deep Transfer Learning, including:

- 1. Zero-shot Learning**
- 2. Few-shot Learning**
- 3. Domain Adaptation**
- 4. Multi-task Learning**

1. Zero-Shot Learning

- Zero-shot learning is the challenge of learning modelling without using data labelling
- This technique is useful for autonomous systems
 - Identify and categorize new objects
- The process goes as following:
 - Model is pre-trained on a set of classes (seen classes)
 - It is asked to generalize to a different set of classes (unseen classes)
 - No additional training data is used

1. Zero-Shot Learning - How does it work

In Zero-Shot Learning, the data consists of the following:

- **Seen Classes**

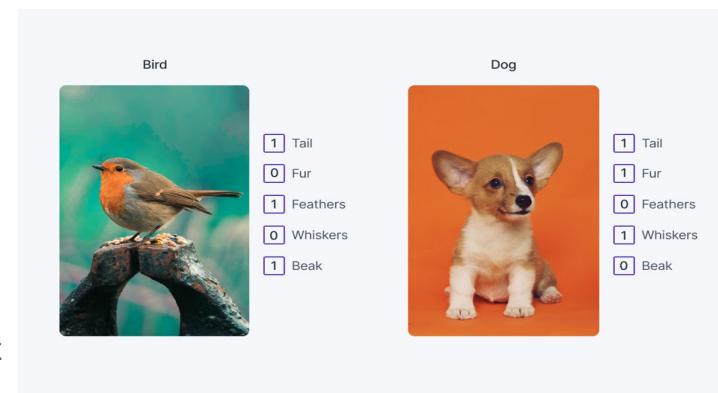
- These are the data classes that have been used to train the deep learning model.

- **Unseen Classes**

- These are the data classes on which the existing deep model needs to generalize.
- Data from these classes were not used during training.

- **Auxiliary Information**

- Some auxiliary information is necessary to solve the Zero-Shot Learning problem.
- It can be descriptions, semantic information, or word embeddings information



1. Zero-Shot Learning - How does it work

Zero-Shot Learning is a two-stage process involving Training and Inference

- **Training**
 - The knowledge about the labeled set of data samples is acquired.
- **Inference**
 - The knowledge previously acquired is extended
 - The auxiliary information provided is used to the new set of classes.

1. Zero-Shot Learning - How does it work

Humans can perform Zero-Shot Learning

- Humans have existing language knowledge base (training)
- Humans need high-level description of a new (unseen) class to make connection between unseen and seen classes.
- Example:
 - If a child is asked to recognise a Yorkshire terrier, he/she may know it is a type of a dog (with added information about it from Wikipedia)



1. Zero-Shot Learning - Challenges

- **Bias**
 - Predicting unseen data samples during the test time as one of the seen classes
- **Domain Shift**
 - Domain shift results when the statistical distribution of the data in the training set (seen classes) and the testing set are significantly different.
- **Hubness**
 - The hubness problem is related to the curse of dimensionality associated with the nearest neighbor search.
 - Some points (called hubs) frequently occur in the k-nearest neighbor set of other points in high-dimensional data.

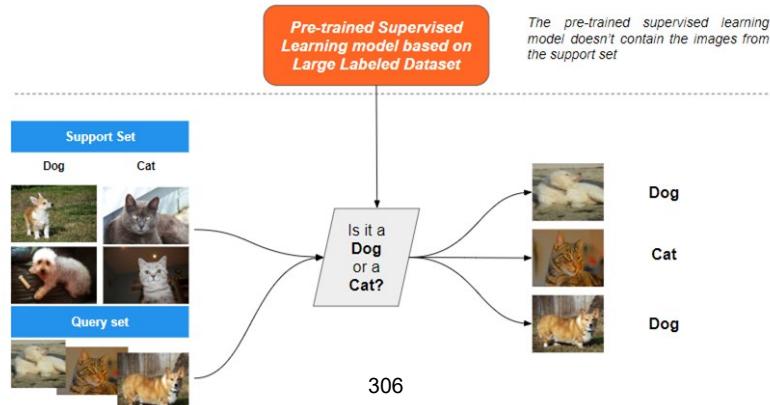


1. Zero-Shot Learning - Applications

- **Image classification**
 - Search Engines can be trained on thousands of classes of images, but still novel objects may be supplied by users
- **Image generation**
- **Object detection**
 - Autonomous navigation applications
- **Image retrieval**
 - Image classification based on sketches to real images
- **Natural Language Processing**
 - Classification of text into topics
- **Audio Processing**
 - Conversion of one speaker to another speaker's voice

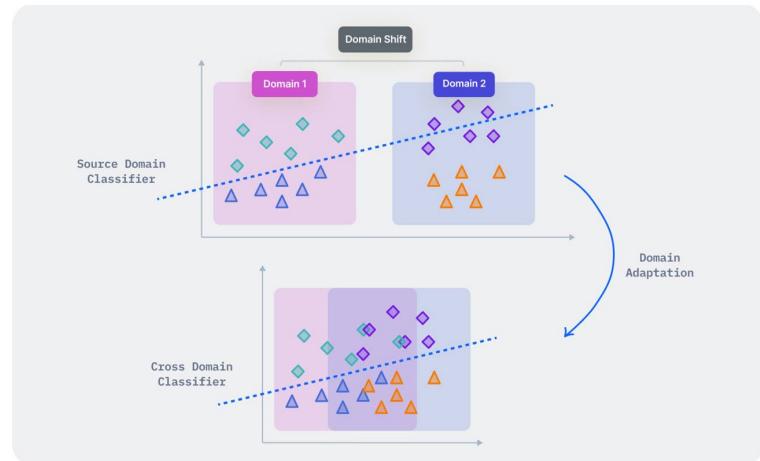
2. Few-Shot Learning

- Few-Shot Learning: a model is trained to classify new classes based on a very limited number of labeled examples
- The goal is to:
 - Leverage prior knowledge learned from other tasks
 - Adapt the model to the new classes with minimal labeled data



3. Domain Adaptation

- **Domain Adaptation is a technique to improve the performance of a model on a target domain**
- Where:
 - Target domain contains insufficient annotated data
 - The knowledge is learned by the model from another related domain with adequate labeled data



3. Domain Adaptation

- Domain Adaptation is special case of Transfer Learning

Key terms:

- **Source Domain**
 - It is the data distribution on which the model is trained using labeled samples
- **Target Domain**
 - It is the data distribution on which a model pre-trained on a different domain is used to perform a similar task
- **Domain Translation**
 - Domain Translation is finding a meaningful correspondence between two domains
- **Domain Shift**
 - A domain shift is a change in the statistical distribution of data between the different domains

3. Domain Adaptation - Types

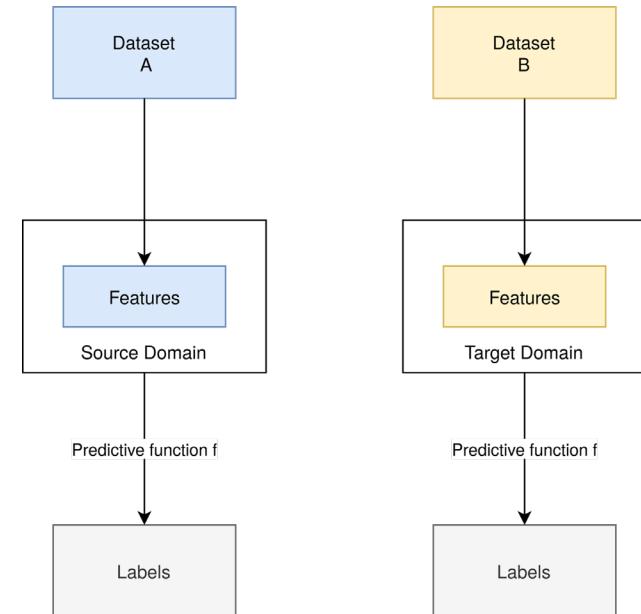
- **Supervised Domain Adaptation**
 - The target domain data is fully annotated
- **Semi-Supervised Domain Adaptation**
 - Only a few data samples in the target domain are labeled
- **Weakly Supervised Domain Adaptation**
 - It is a problem setting wherein only “weak labels” are available in the target domain
- **Unsupervised Domain Adaptation**
 - Any kind of labels (weak/hard) for the target domain data are entirely missing

3. Domain Adaptation - How to apply

1. Dataset is similar but not the same
2. Features of the model stays the same
3. Function of the model stays the same

Example: cat/dog dataset

- **Source dataset** contained only **poodles** and **black cats**
- **Target dataset** still contains cats/dogs but **schnauzers** and **white cats**

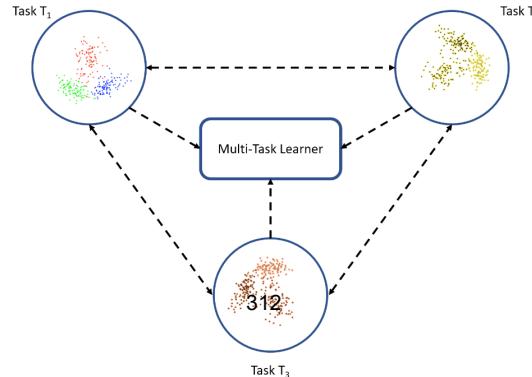


3. Domain Adaptation - Examples

1. An algorithm trained on **newswires** have to **adapt** to a new dataset of **biomedical** documents
2. **A spam filter** (trained on a certain group of email users) must **adapt** to a new target user when deployed
3. Applying **AI diagnostic** algorithms (trained on **previous diseases**) on data associated with the COVID-19.
4. **Adapt** corpus of **movie reviews** (labeled as positive or negative) on different from a corpus of **product-review** sentiments.

4. Multi-task Learning

- Several tasks **from the same domain** are learned **simultaneously** without distinction between the source and targets.
- The model is trained on a set of tasks:
 - a. Tasks use the same feature extractor
 - b. Separate task-specific classifiers are used for each task



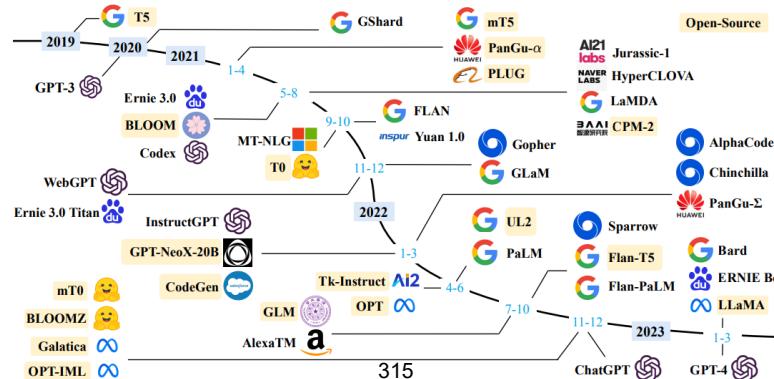
4. Multi-task Learning - Example

- Create network that classify an input face image as **male** or **female**
- At the same time it can predict its **age**
- These two tasks are related
 - One is binary
 - The other is regression task
- Learning one should enhance learning the other

DELETED/BACKUP

Train the Predictive Model

- **Model Architecture**
 - Choose a suitable model architecture for the task.
- **Training Process**
 - Input prepared data to the model and adjust model parameters iteratively.



Fine Tune for Harmful (GAN)

Fine Tune for Useful (RLHF)

How to use pre-trained models

There are three ways how to use a pre-trained model:

- **Prediction/Classification**
- **Feature extraction**
- **Fine-tuning**

How to use pre-trained models - Prediction

Pre-trained models could be used directly for **Prediction**:

1. Download model
2. Use it immediately for classification/prediction
3. Examples:

- ResNet50 - classify images
- Bert - entity extraction
- ChatGPT

Example with ResNet50

```
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input,
decode_predictions

import numpy as np
model = ResNet50(weights='imagenet')
img_path = cat.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)
preds = model.predict(x)
print('Predicted:', decode_predictions(preds, top=3)[0])
```

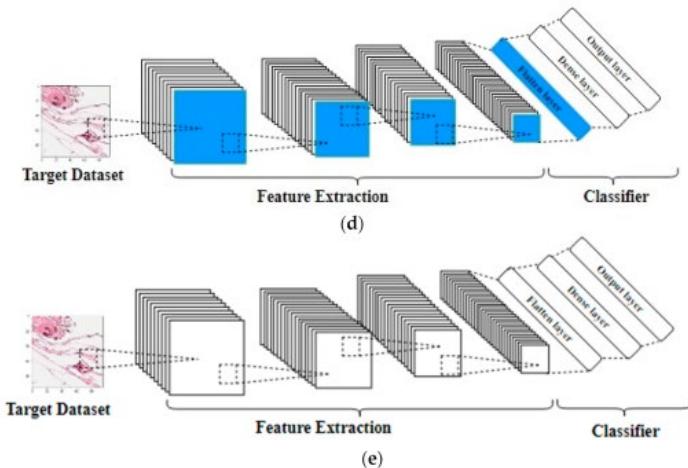
How to use pre-trained models - Feature Extraction

Pre-trained models could be used for **Feature Extraction**:

1. Use the pre-trained model for **pre-processing data**
2. **Get essential features** (e.g., use embedding)
3. Use newly created features to build classifier

Example:

- Use pre-trained word embeddings as features to calculate cosine similarity between two documents



Popular pre-trained models for generative AI applications

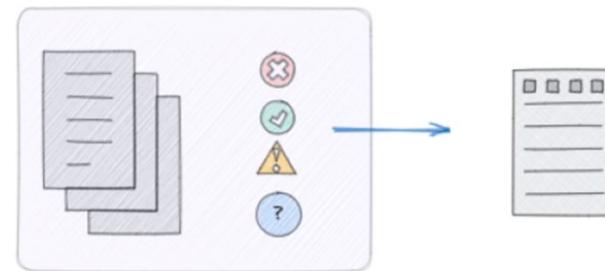
- **GPT-3**
 - Generative Pre-trained Transformer 3 by OpenAI
 - Comprehends human language prompts and generates human-like text
 - Can be fine-tuned for tasks like translation, question-answering, and summarization
- **DALL-E**
 - Language model by OpenAI
 - Generates images from textual descriptions
 - Trained on a large dataset of images and descriptions
- **BERT**
 - Bidirectional Encoder Representations from Transformers by Google
 - Used for various tasks, including question answering, sentiment analysis, and language translation
 - Trained on a large amount of text data and can be fine-tuned for specific language tasks

Popular pre-trained models for generative AI applications

- **StyleGAN**
 - Style Generative Adversarial Network by NVIDIA
 - Generates high-quality images of animals, faces, and objects
- **VQGAN + CLIP**
 - Generative model by EleutherAI
 - Combines a generative model (VQGAN) and a language model (CLIP) to generate images based on textual prompts
 - Uses a large dataset of images and textual descriptions
- **Whisper**
 - Speech recognition model by OpenAI
 - Trained on diverse audio data
 - Multi-task model capable of tasks like multilingual speech recognition, speech translation, and language identification

Preparation and structuring the data

- Labeling data
- Formatting the data into suitable format for the model
- Split data into:
 - **Train**
 - **Test**
 - **Dev/Evaluation**

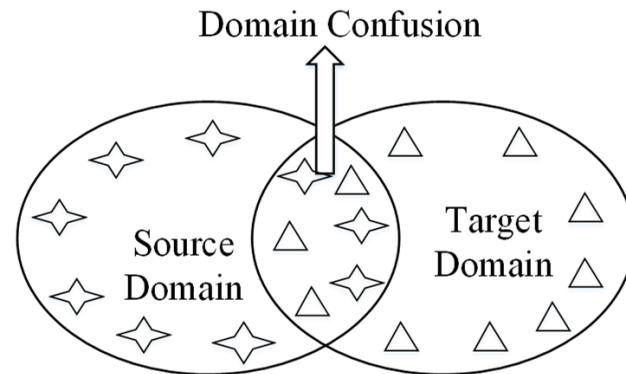


4. Domain Confusion

- Train a model to minimize classification loss and **confuse two domains**
- The goal is:
 - Add an objective to the model at the source
 - Encourage similarity with the target
 - Do it by confusing the source domain itself

4. Domain Confusion

- Domain confusion loss is used to confuse the high-level classification layers
- It is done by **matching the distributions of the target and source domains.**



Chatbots

Chapter 9

Agenda

1. Chat Bot Basics
2. Building LLM-Based Chat Bots

Chat Bot Basics

Chatbot - Architectures

There are two main types of chatbots:

- **Retrieval-Based** Chatbots
- **Generative** Chatbots



Retrieval-Based Chatbots

Retrieval-based chatbots generate response by selecting the most suitable pre-defined response.

Advantages:

- **Faster response** times due to predefined responses.
- Requires **less complex training** compared to generative models.

Challenges:

- **Limited creativity** in responses.
- Struggles with handling inputs that don't match predefined patterns.

Generative Chatbots

- Generative chatbots create responses from scratch using NLP techniques.
- These chatbots are powered by LLMs

Advantages:

- Produces more **diverse responses**
- Can handle a wide range of user inputs

Challenges:

- Requires substantial training data
- Requires big computational resources
- May produce incorrect or nonsensical answers

Enhancing Chatbot Performance with LLMs

By using LLMs in building Chatbots, you can have:

- **Enhanced Efficiency**
 - LLM AI handles complex queries and learns from interactions.
- **Improved Customer Experience**
 - LLM enables personalized responses and accurate solutions.
 - Chatbots understand context and enhance user experience.
- **Increased Accessibility**
 - Chatbots understand various languages and dialects.
 - Chatbots can answer customer questions outside of call center business hours

Enhancing Chatbot Performance with LLMs

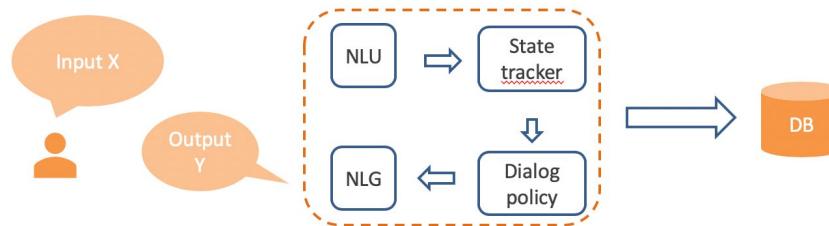
By using LLMs in building Chatbots, you can have:

- **Greater Versatility**
 - LLM continuous learning lets chatbots handle diverse topics.
- **Advanced Personalization**
 - Chatbots consider user history and preferences.
- **Superior Integration Capabilities**
 - LLM enables better integration with systems and platforms.

Traditional Chatbot Functionality

The following are the core Chatbot functionalities:

- Natural language understanding (NLU)
- Natural language generation (NLG)
- Context-aware conversation handling



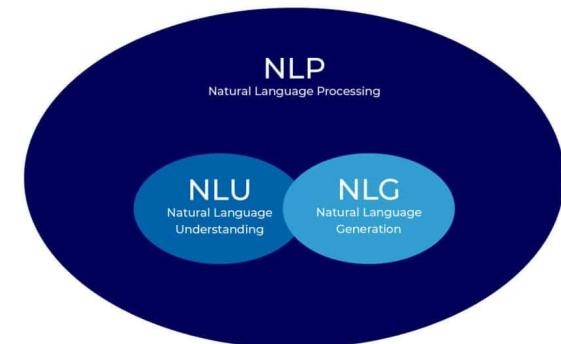
Natural language understanding

Defining NLU

- NLU is the capability to comprehend the meaning of user input.
- Focused on machine reading comprehension and understanding text.

Comprehending Text

- NLU enables chatbots to grasp the meaning behind text.
- Classifies user input into appropriate intents.



Natural language understanding

Components of NLU

- Dialogue Agent
- Semantic Parsing
- Question Answering
- Sentiment Analysis
- Summarization
- ..

Natural language generation

Defining NLG

- NLG is software that produces human-readable text.
- Creates text for documents, explanations, and more.

Symbiotic Human-Machine Systems

- NLG techniques combine human and machine knowledge.

NLG Inputs and Outputs

- Non-linguistic information as input (database)
- Textual output for:
 - Documents
 - Reports
 - Explanations
 - etc.

Context-aware conversation handling

- **Importance of Context**
 - Context enhances chatbot interactions
 - Chatbots consider ongoing dialogue for responses
- **Retaining Contextual Memory**
 - LLM-based chatbots maintain dialogue history
 - Contextual memory enables relevant follow-ups and references
- **Personalized Engagement**
 - Context-awareness fosters personalized user experiences

Building LLM-based Chat Bots

How to build a Chatbot with a LLM

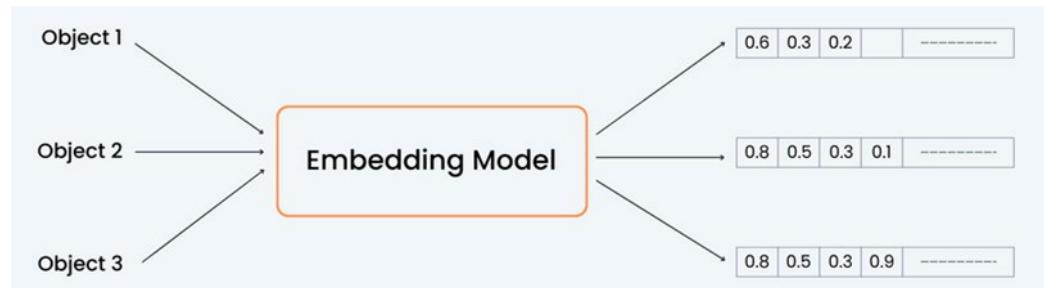
STEP 1: Organizing Knowledge Base

- Break knowledge base into manageable text chunks
- Gather your private data from
 - Pdfs
 - Confluence
 - Conversations with clients
 - ...
- Store chunks with clear boundaries for easier querying

How to build a Chatbot with a LLM

STEP 2: Text into Vectors

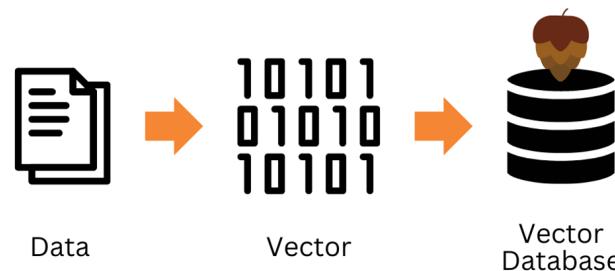
- Use embedding model to convert text chunks into vectors
- Train the model on a substantial text corpus
- Ensure fixed-length vectors for easy storage and retrieval



How to build a Chatbot with a LLM

STEP 3: Store Vector Embeddings

- Save vectors from the embedding model in a **Vector Database**
- Efficient storage and retrieval of vectors
- Index vectors for keyword-based searching



How to build a Chatbot with a LLM

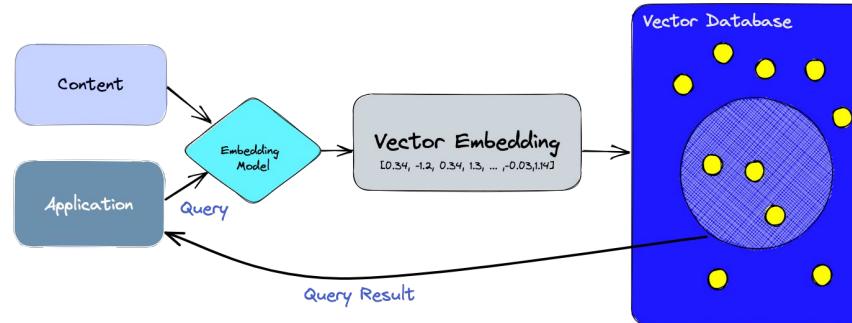
STEP 4: Preserve Original Text

- Store original text corresponding to each vector
- Vital for retrieving context during queries
- Store separately from vector embeddings

How to build a Chatbot with a LLM

STEP 5: Embed the Question

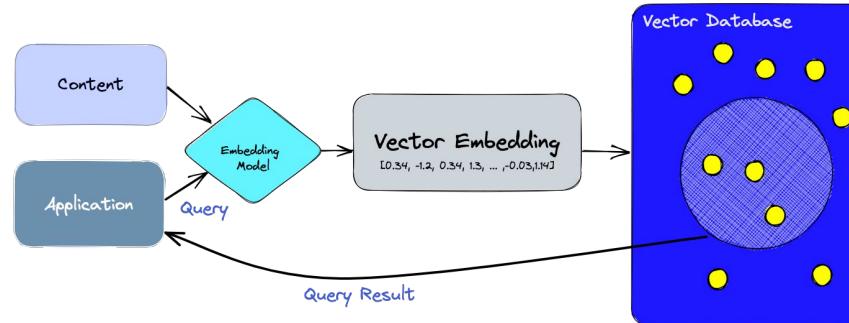
- Transform questions into vector representations
- Use the same embedding model for consistency
- Align the question vector with the context vector



How to build a Chatbot with a LLM

STEP 6: Perform a Query

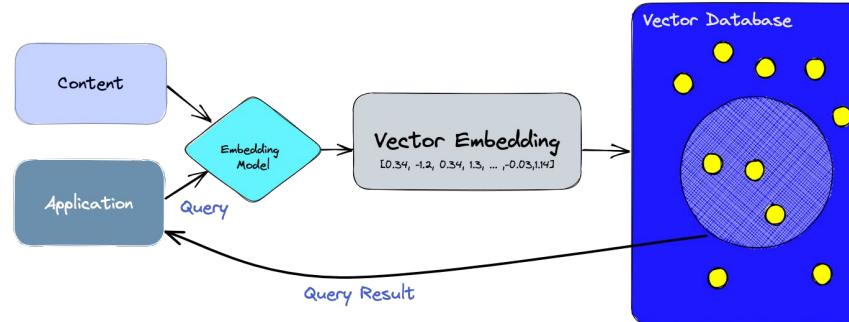
- Query Vector Database using the question's vector.
- Retrieve relevant context vectors for assistance.
- Context vectors should match the question's representation.



How to build a Chatbot with a LLM

STEP 7: Retrieve Similar Vectors

- Conduct an Approximate Nearest Neighbor (ANN) search.
- Find the most similar vectors to the query embedding.
- Extract relevant information from selected context vectors.



How to build a Chatbot with a LLM

STEP 8: Map Vectors to Text Chunks

- Associate retrieved vectors with corresponding text chunks.
- Link numerical representations to actual content.
- Maintain mapping in a separate database or file system.

How to build a Chatbot with a LLM

STEP 9: Generate the Answer

- Pass question and context chunks to the LLM via a prompt.
- Instruct LLM to generate the answer using provided context.
- LLM will generate a relevant natural language answer

Building Generative AI Applications (part 1)

Chapter 10

Agenda

- Application Design Building Blocks
- Use Cases of LLM Based Application
- Prompt Engineering Basics
- Prompt Templates
- RAG with Llama Index

Application Design Building Blocks

Text Generation Applications Basics

- Called “generative” because new content is generated, unlike classical machine learning systems
- Uses LLM’s like GPT-3 and GPT-4 (OpenAI), Llama (Meta), Free Dolly (DataBricks) capable of generating human-like text.
 - The underlying models learn from vast amounts of text data and can produce text that is contextually relevant and linguistically coherent.
 - The application must send correct API calls (prompts) to the API to cause the LLM to generate the exact, desired output.

LLM Based Generative AI Applications

- LLM based apps generate new text based on user inputs
- Recent advances in LLM's have significantly improved the performance and capabilities of these apps over legacy chatbots
- These include coding assistants (code is just another language!)
- The applications query the LLM, and build an application around the response

Text Generation

- Opportunities:
 - Automating content creation: Can save time and efforts for content creators.
 - Personalized content: Can generate content tailored to individual user preferences.
- Challenges:
 - Maintaining coherence over long texts: AI can sometimes lose coherence in long pieces of text.
 - Avoiding harmful or biased outputs: Ensuring AI doesn't generate offensive, inappropriate, or biased content.

Text Generation Advances

- Larger LLM's show an emergent capability to handle longer, more complex questions
- Local or off-line models provide security and privacy benefits
- Newer LLM architectures show improved performance with smaller models
- Newer LLM's also show increased speed and lower resource requirements

LLM Based Generative AI Applications

- The current, most common techniques for building applications around LLM's include:
 - **Prompt Engineering**
 - **Retrieval Augmented Generation (RAG)**
 - **Fine Tuning**

Prompt Engineering

- Prompt engineering is the art and science of designing effective and efficient prompts to elicit specific, targeted responses from language models.
- It involves understanding the nuances of language, context, and model behavior
- The goal is to optimize user interactions and achieve desired responses from the LLM

Fine Tuning

- Fine Tuning creates a derivative model based on the original LLM
- The derivative learns new knowledge and modifies part of the neural network of the original LLM
- Requires a large amount of text data to work
- May suffer from Catastrophic Forgetting (overwriting previous LLM's knowledge) and/or underfitting (not learning the new data)
- New LLM runs in same environment as original LLM

Data Considerations for Fine Tuning

- **Privacy and Sensitivity**
 - Generated data must not violate privacy regulations
 - Disclose sensitive information
- **Data Transformation and Cleaning**
 - automate certain data transformation and cleaning tasks
 - domain expertise required

Synthetic Data for Fine Tuning

- **Data Augmentation and Synthesis**
 - Effectively generate synthetic data
 - Ensure generated data retains statistical characteristics of original data
 - Avoid bias and unrealistic patterns
- **Quality Control and Validation**
 - Generated data should be **validated** and **compared against real-world data**
 - Verify accuracy and usefulness
 - Combination of LLM-generated and real data for training robust models

Considerations for Generative Code LLM Tuning

- **Code Quality and Consistency**
 - Generate code snippets
 - Code may not cover all possible edge cases or error scenarios
 - Review is necessary
- **Ethical and Legal Considerations**
 - Generated content must not infringe on intellectual property rights

Retrieval Augmented Generation (RAG)

- RAG relies on current data to generate a response in 5 steps

Retrieval

- Get current documents from external sources

Vector Indexing:

- the indexing step encodes the retrieved documents for similarityty comparison

Similarity Scoring:

- indexed documents are compared for similarity.

Ranking and Retrieval

- documents are ranked for relevance based on similarity

Generation

- New text is generated by the LLM and is informed by the most relevant documents

LLM Embeddings of Reference Documents

- Used by Vector Databases
- Used by RAG

Vector Indexing Databases

- RAG is limited to text data
- Vector indexed databases can contain non-text data
- Because they contain a curated dataset vector indexed databases tend to be more purpose built and employed for specialty applications

Use Cases of LLM Based Application

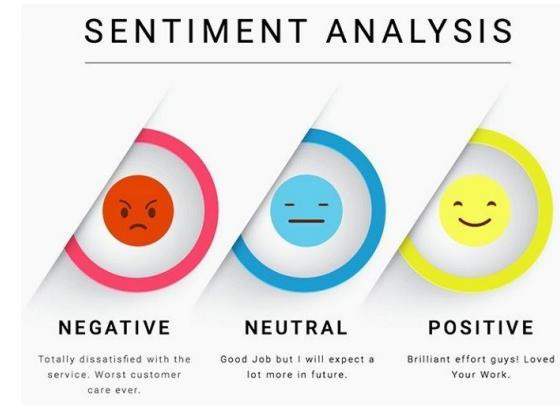
Sentiment Analysis

- Sentiment Analysis is the process of determining the **sentiment** or emotional tone of a piece of text, whether it's:
 - Positive
 - Negative
 - Neutral
- Large Language Models (LLMs) like GPT-3.5 offer a powerful foundation for sentiment analysis due to their deep understanding of language and context.

Sentiment Analysis

Steps for Fine-Tuning:

- **Start with a pre-trained LLM**
 - Examples: GPT-3.5., Bert
- **Gather a dataset of labeled text**
 - Examples with sentiment labels (positive, negative, neutral)
- **Fine-tune the LLM**
 - Use collected sentiment dataset
- The LLM adjusts its internal parameters to learn sentiment-related patterns.



Named Entity Recognition (NER)

- **Named Entity Recognition (NER)** is a task that involves identifying entities in text such as:
 - Names
 - Companies
 - Dates
 - Locations
 - and more
- Large Language Models (LLMs) like GPT-3.5 can be fine-tuned for NER to identify and classify entities within text.

Named Entity Recognition (NER) - Fine tuning steps

1. Data Collection

- Gather a dataset with annotated examples containing entities like names, locations, dates, etc.
- Examples: "John visited Paris on 2022-05-15."

2. Fine-Tuning

- Start with a pre-trained LLM
 - Provide the model with annotated data (next slide)
- The model adjusts its weights to better identify and classify entities during the training

3. Inference

- After fine-tuning, use the model to predict entities in new text.
 - Input: "**Apple** announced a new product launch in **Cupertino**."
 - Output: {"**Apple**": "ORG", "**Cupertino**": "LOC"}

Named Entity Recognition (NER) - Data input

```
{  
    'id': '0',  
    'tokens': ['EU', 'rejects', 'German', 'call', 'to', 'boycott', 'British',  
    'lamb', 'in', '2020'],  
    'ner_tags': ["ORG", "O", "MISCELLANEOUS", "O", "O", "O",  
    "MISCELLANEOUS", "O", "O", "O", "DATE"],  
    'sentence': 'EU rejects German call to boycott British lamb in 2020'  
}
```

Text Summarization

What is Summarization?

- Summarization is the process of condensing a large piece of text into a shorter version while retaining its key information.

Importance of Summarization:

- **Information Overload**
 - In the digital age, there's an abundance of information. Summaries help in quickly grasping content.
- **Time Efficiency**
 - Readers save time by getting a quick overview before deciding to delve deeper.
- **Content Sharing**
 - Summaries facilitate sharing insights and ideas in a more accessible manner.

Text Summarization

Use Cases:

- **News Aggregation**
 - Extracting headlines and main points from various news articles.
- **Academic Research**
 - Condensing lengthy research papers for quick understanding.
- **Business Reports**
 - Creating concise summaries of market analysis and financial reports.

Text Summarization

Fine-Tuning Process:

- **Dataset Preparation**
 - Curate a dataset of source texts and corresponding summaries.
- **Model Initialization**
 - Start with a pre-trained LLM.
- **Task-Specific Training**
 - Fine-tune on summarization data
- **Validation and Tuning**
 - Monitor model performance and fine-tuning parameters.
- **Evaluation**
 - Assess the model's summarization quality using metrics like ROUGE

Question Answering (QA)

What is Question Answering?

- Question Answering (QA) is a NLP task where a model answers questions based on a given context or passage.
- It involves:
 - **Understanding** the context
 - **Extracting** relevant information
 - **Generating** accurate answers

Question Answering (QA)

Importance of Question Answering:

- **Information Retrieval**
 - Quickly find specific information from large volumes of text.
- **Conversational Agents**
 - Power chatbots and virtual assistants with the ability to answer user queries.
- **Learning and Research**
 - Extract knowledge and insights from text for educational and research purposes.

Question Answering (QA)

Use Cases:

- **Customer Support**
 - Providing instant answers to user queries on websites or platforms.
- **Educational Resources**
 - Assisting students by answering questions from textbooks or research materials.
- **Data Analysis**
 - Extracting insights from lengthy reports and documents.

Chatbots

What are Chatbots?

- Chatbots are AI-powered conversational agents that simulate human-like interactions through text or speech.
- Chatbots offer:
 - **Information**
 - **Assistance**
 - **Engagement**

Chatbots

Importance of Chatbots

- **24/7 Availability**
- **Efficiency and Speed**
 - Chatbots can handle multiple conversations simultaneously,
- **Scalability**
 - Chatbots can easily handle a growing number of users
- **Automation**
 - Streamline repetitive tasks and handle routine queries
- Many other

Chatbots

Use Cases:

- E-commerce
 - Help customers find products, answer questions, and facilitate purchases.
- Healthcare
 - Provide medical information, appointment scheduling, and symptom assessment.
- Banking
 - Assist with account inquiries, transactions, and financial advice.

Document classification

What is Document Classification?

- Document classification involves categorizing documents into predefined classes or categories based on their content, features, or characteristics.

Importance of Document Classification:

- **Efficient Information Retrieval**
- **Decision-Making Support**
- **Search Optimization**
- **Automated Workflows**

Document classification

Use Cases:

- **Legal Industry**
 - Classifying legal documents for efficient case management
- **Healthcare Sector**
 - Categorizing patient records and medical reports
- **Business Intelligence**
 - Organizing financial reports and market analyses
- **Content Management**
 - Categorizing articles and blogs for easy access
- **Customer Service**
 - Classifying support tickets for timely resolution
- **Many other**

Virtual Assistant

- AI-powered virtual assistants can understand human speech, recognize patterns, and respond with relevant information or actions.
- Examples include Siri, Google Assistant, and Alexa, which help users with tasks like setting reminders, answering questions, or controlling other smart devices.

Virtual Assistant

- Techniques:
 - Natural Language Processing (NLP): Used to understand and generate human language.
 - Speech Recognition: Transcribes

PROMPT ENGINEERING BASICS

Definition

- **Art and Science:** Prompt engineering is both an art and a science that involves crafting effective prompts to get desired responses from language models.
- **User-Model Interaction:** It focuses on the interaction between the user and the model, tailoring prompts for better accuracy and relevancy.

Purpose

- **Optimization:** The goal is to optimize how the model interprets and responds to user queries, improving both performance and user experience.
- **Broad Applications:** Useful in a wide range of applications from customer service bots to data analysis tools.

Prompt Types

- **Open-Ended vs. Closed-Ended:** Prompts can be designed to elicit either expansive responses (open-ended) or specific, targeted answers (closed-ended).

Context

- **Attention Mechanism:** LLM's use an attention mechanism to weigh the importance of different words in the input when making predictions.
- This means that the additional content that gives context is critical for engineering the desired LLM responses.
- **This is one of the most important points to remember for Prompt Engineering!**

Word Choice

- **Word Choice:** The choice of words and phrasing can significantly impact the model's response.
- Remember, LLM's predict the next sequence of words. The way the prompt is phrased will impact the response!

Formatting

- **Formatting:** Structuring the prompt with lists, bullet points, or special characters can influence the output.
- Formatting is also a great way to separate the instructions from the contextual part of the prompt

Prompt Pitfalls

- Vague Prompts
- Leading Prompts

Vague Prompts

- **Lack of Clarity:** Vague prompts can result in ambiguous or off-target responses from the model.
- **Example:** Asking “Tell me about it” could yield information that’s not directly relevant to the user’s intent.

Leading Prompts

- **Biasing Output:** Prompts that are too leading can inadvertently guide the model to produce biased or skewed responses.
- **Example:** Asking “Why is X the best option?” may result in a one-sided view, ignoring potential drawbacks.

PROMPT Templates

What Are Prompt Templates?

- **Definition:** Structured and reusable formats or skeletons to elicit specific types of responses from a language model.
- **Function:** Allows fine-tuning of the model's behavior without retraining.

EXAMPLE: "Translate the following {language} text to English:\n{text_here}"

Examples of Prompt Templates

- **Translation:** "Translate the following [source_language] text to [target_language]: [text_here]".
- **Summary:** "Provide a concise summary of the following article: [article_text_here]".
- **Question Answering:** "Given the text [context_here], answer the following question: [question_here]".

RAG with Llama Index

Need for Additional Content

Challenges in LLMs:

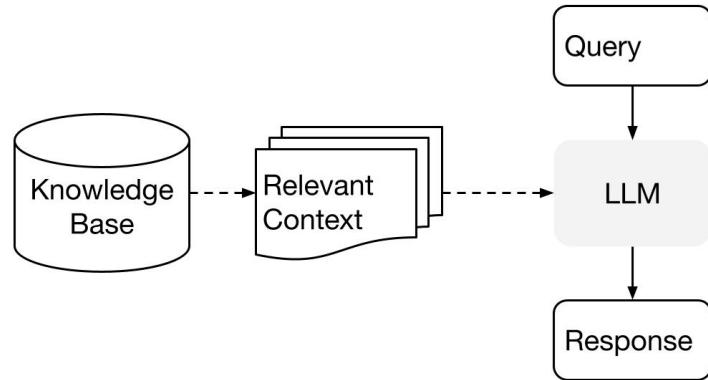
- Bias and Fairness
 - LLMs often inherit biases from the data they are trained on, which can result in biased language generation.
- Ethical Concerns
 - Generation of misinformation information, data privacy.
- Resource Intensiveness
 - Training and fine-tuning large LLMs require big computational resources.
- **Lack of domain-specific knowledge**
 - **RAG can help solve this issue**

RAG

Retrieval augmented generation (RAG) is a paradigm for augmenting LLM with custom data.

It has two stages:

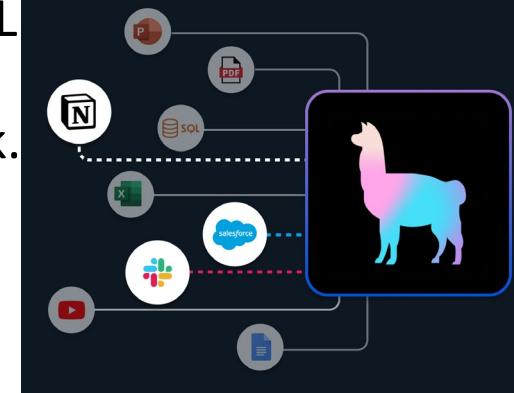
- Indexing stage
 - Preparing a knowledge base, and
- Querying stage
 - retrieving relevant context from the knowledge base



LlamaIndex provides the toolkit for making both steps easy.

What is Llama Index?

- Llama Index is a library for RAG
- Llama Index is a user-friendly, flexible **data framework**
- It connects private, customized data sources to LLMs.
- It acts as a managed interaction between users and LLMs.
- Builds various types of indexes depending on the task.



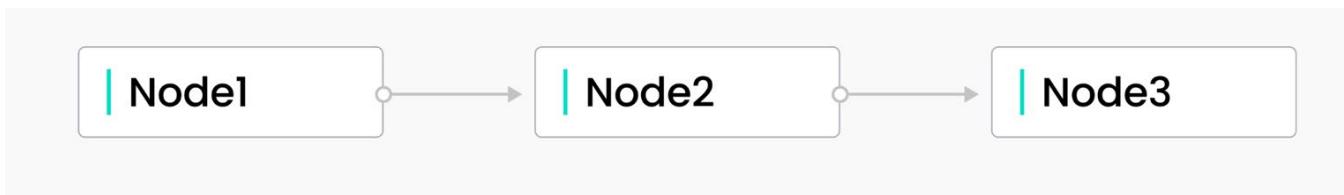
Types of Indexes in Llama Index

Types of Indexes

- **List Index**
 - Sequential indexing of input data.
- **Vector Store Index**
 - Stores nodes as vector embeddings for similarity searches.
- **Tree Index**
 - Builds a tree structure from input data for efficient querying.
- **Keyword Index**
 - Maps keywords to nodes for targeted searches.

Types of Indexes in LlamaIndex - List Index

- Represents data in a list-like structure.
- Input data is divided into sequential nodes.
- The nodes are queried sequentially
- It supports advanced queries using keywords or embeddings.



Types of Indexes in LlamaIndex - Vector Store

- A Vector Store is designed to **store nodes as vector embeddings**.
- Storage options include:
 - local storage
 - purpose-built vector databases like Milvus.
- When queried, LlamaIndex identifies the **top_k** most similar nodes.
- Best suited for workflows that involve **comparing texts for semantic similarity** using vector search.

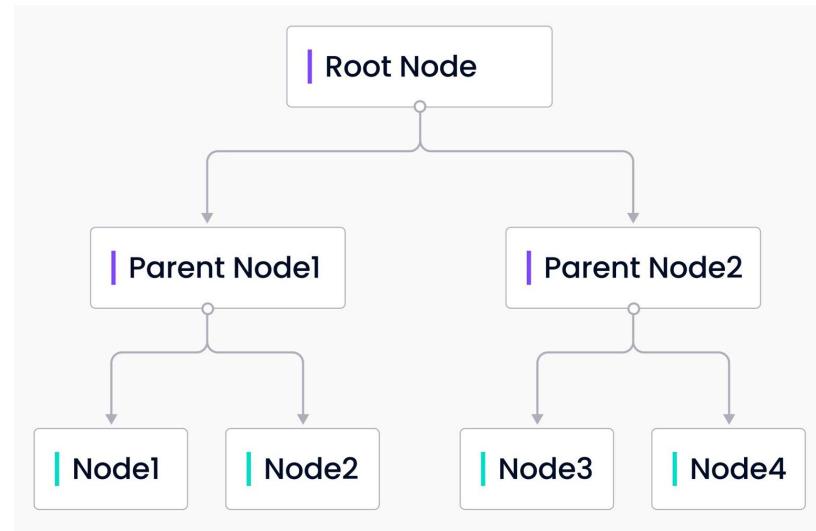


Types of Indexes in LlamaIndex - Tree Index

A Tree Index constructs a hierarchical tree structure from input data.

Construction Process:

- Built **bottom-up** from original input data chunks (leaf nodes).
- Parent nodes serve as summaries of the leaf nodes.
- GPT is utilized by LlamaIndex to generate node summaries and build the tree.



Types of Indexes in LlamaIndex - Tree Index

Querying Flexibility:

- Responses can be generated by:
 - Traversing from root to leaf nodes
 - Directly from selected leaf nodes

Efficiency Advantages:

- Efficient for querying lengthy text passages.
- Enables extraction of information from different parts of the text.

Example Use Case:

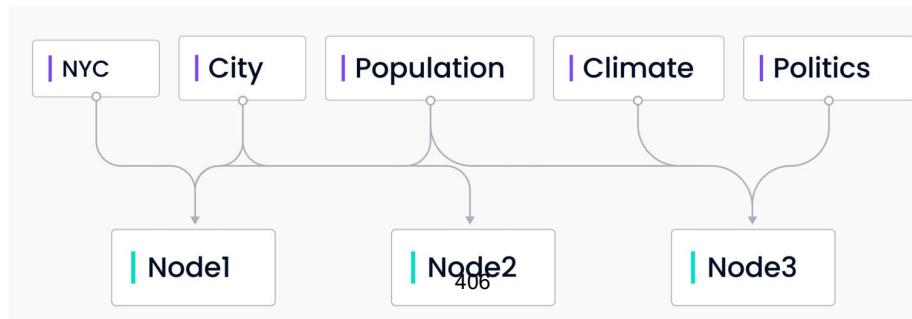
- Retrieve information from long lengthy document (from specific sections)

Types of Indexes in LlamaIndex - Keyword Index

A **Keyword Index** functions as a mapping of keywords to nodes containing those keywords.

Mapping Complexity:

- Utilizes a many-to-many mapping approach.
- Each keyword can point to multiple nodes, and each node can be associated with multiple keywords.



Types of Indexes in LlamaIndex - Keyword Index

Query Process:

- During query time, keywords are extracted from the user's query.
- Only nodes mapped to the extracted keywords are queried, streamlining the search process.

Efficiency Benefits:

- Effective when querying extensive datasets for specific keywords.
- Ideal when the user's query focuses on particular keywords

Example Scenario:

- Consider a database of health care-related documents.
- The Keyword Index enables quick retrieval of documents related to "COVID-19" when the user is specifically interested in that topic.

Differences in Index types

	Indexing	Retrieval	Benefits	Drawback
Knowledge Graph Indexing (Tree)	Builds a knowledge graph with keywords and relations between them.	Uses the knowledge graph to find relevant documents	Preserves relations between nodes	Building the knowledge graph can be time-consuming
Keyword Table Index	Extracts keywords from indexed documents	Generates keywords from the question, searches for relevant documents based on these keywords	Provides keyword-based retrieval and indexing. Allows for easy inspection of generated keywords	Indexing can be expensive and slow as it requires sending every document to the LLM to generate keywords.
List Index	Concatenates all indexed nodes/documents	Passes the entire concatenated text to the LLM for generating answers.	Suitable for a few questions to answer with a small number of documents.	Can be expensive (as you pay per token) and sending all documents to the LLM might not be cost-effective.
Vector Store Index	Creates numerical vectors from text using word embeddings.	Retrieves relevant documents based on the similarity of vectors 408	Efficient and cost-effective indexing and retrieval.	The quality of answers depends on the quality of embeddings