

8716 Generative AI Overview



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: Building Generative AI Applications (part 1) ...81
- Ch. 4: Case Studies and Real-World Applications ...141
- Ch. 5: Building Generative AI Applications (part 2) ...207

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

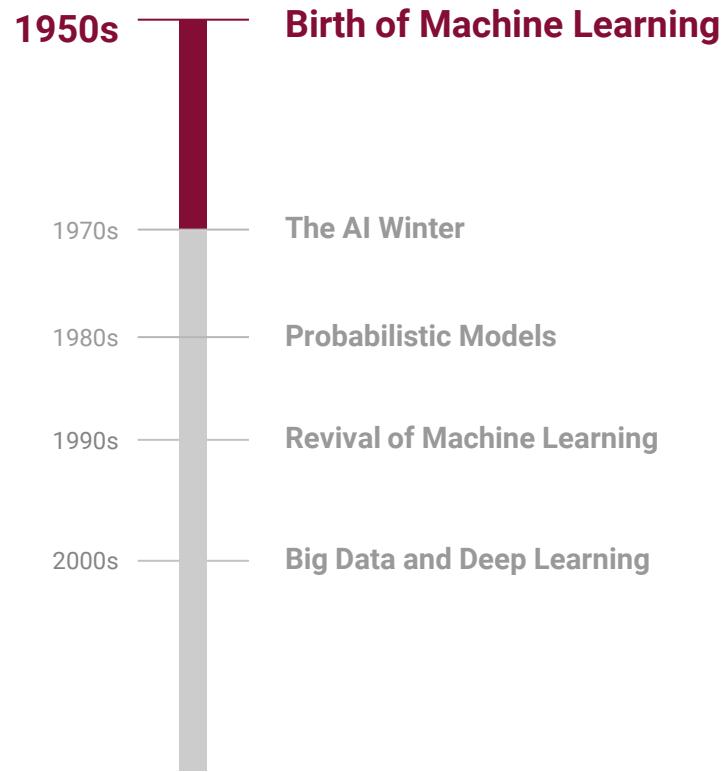
- **What is ML?**
 - **Machine learning** (ML) is a subset of Artificial Intelligence (AI) that allows computers to **learn from data without any programming intervention**.
 - ML algorithms **use data to identify patterns** and learn from them, making it possible to automate tasks and predict outcomes.

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

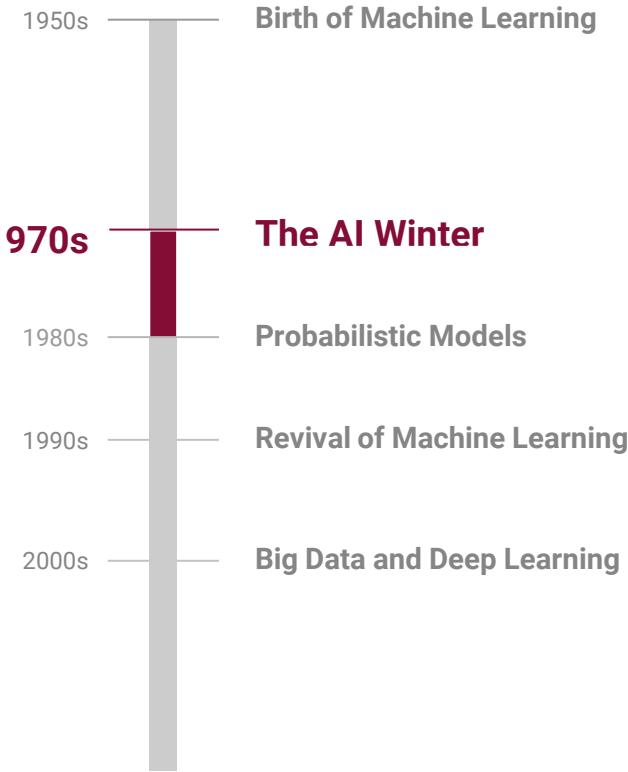


How Machine Learning Led to Generative AI

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

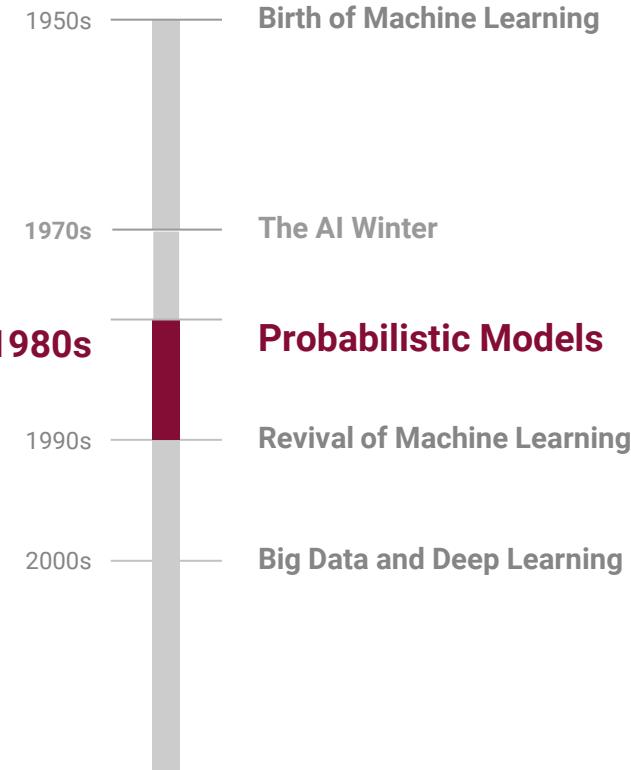


How Machine Learning Led to Generative AI

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.**

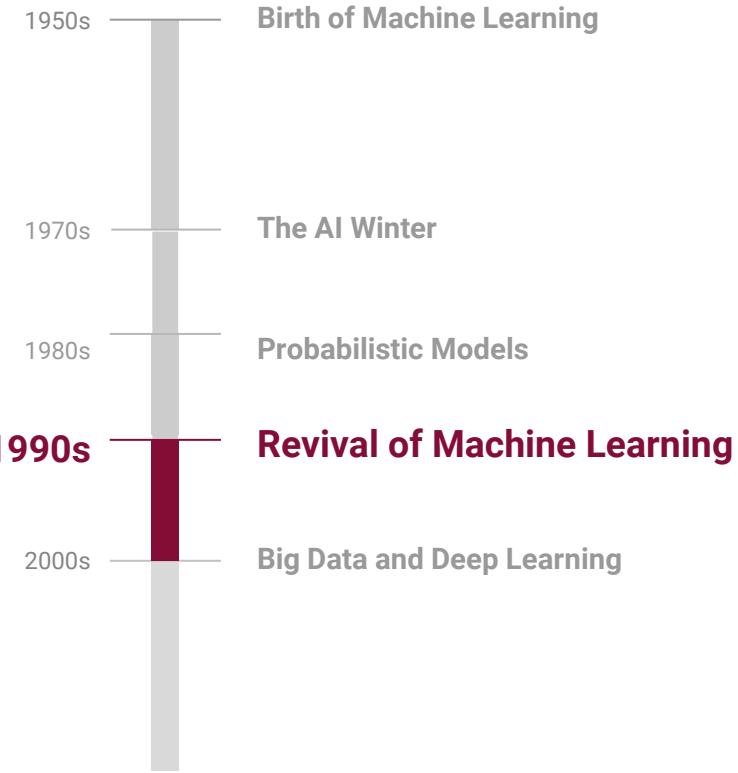


How Machine Learning Led to Generative AI

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.

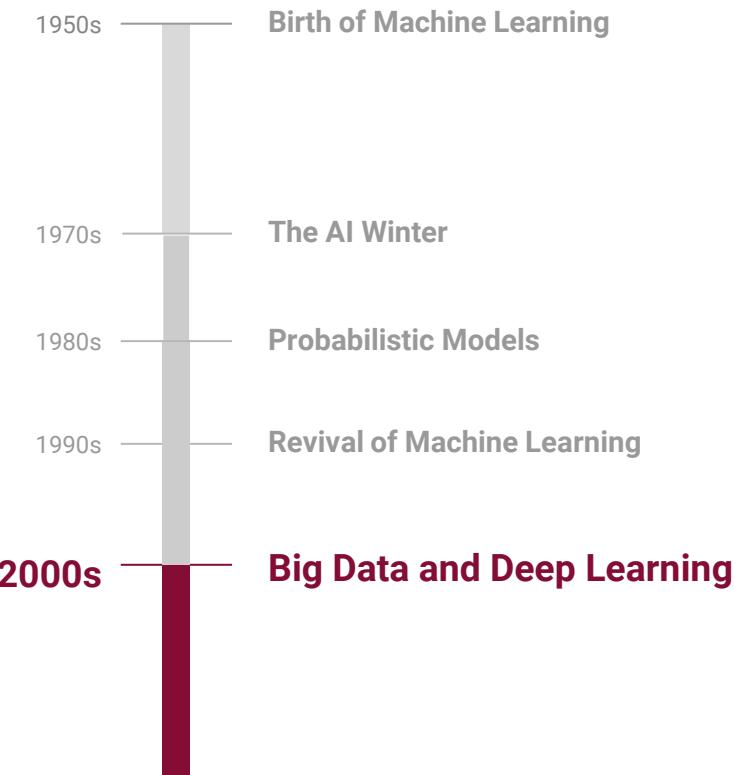


How Machine Learning Led to Generative AI

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**

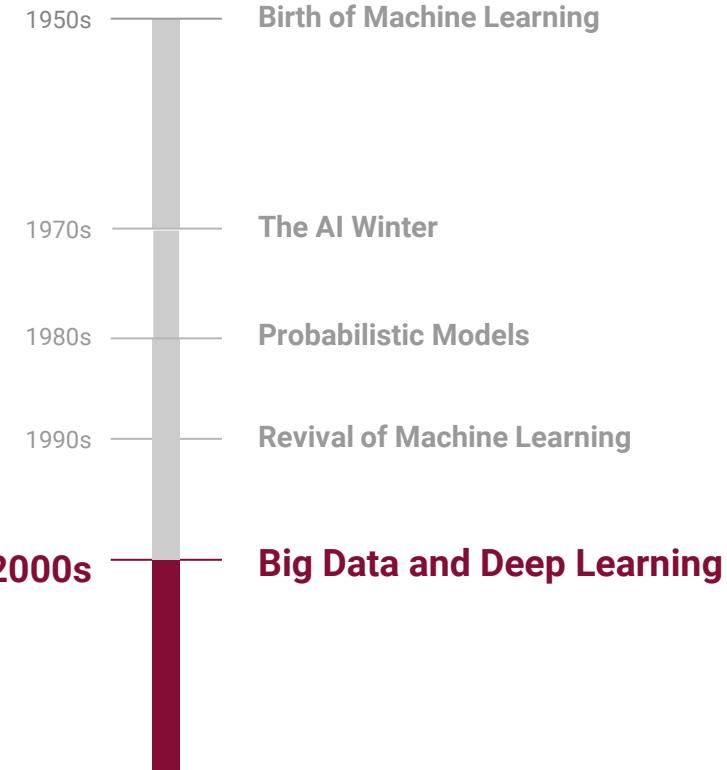


How Machine Learning Led to Generative AI

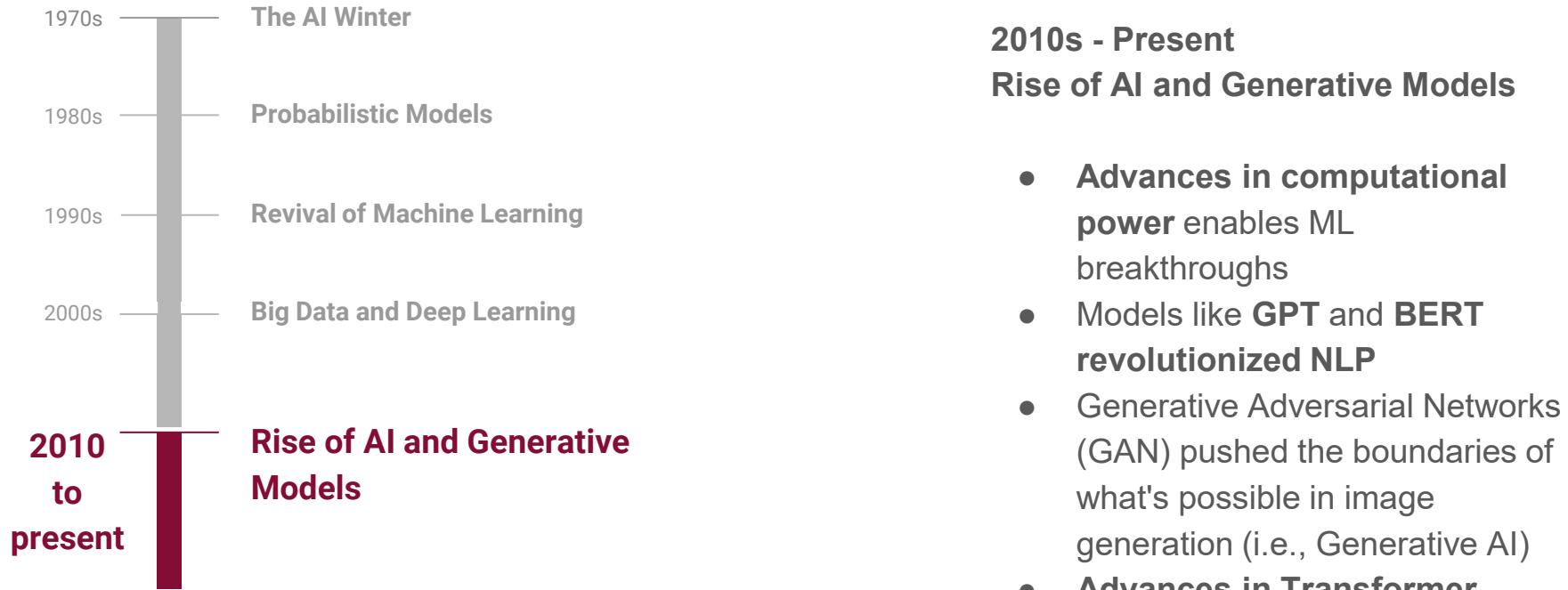
2000s - 2010s

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

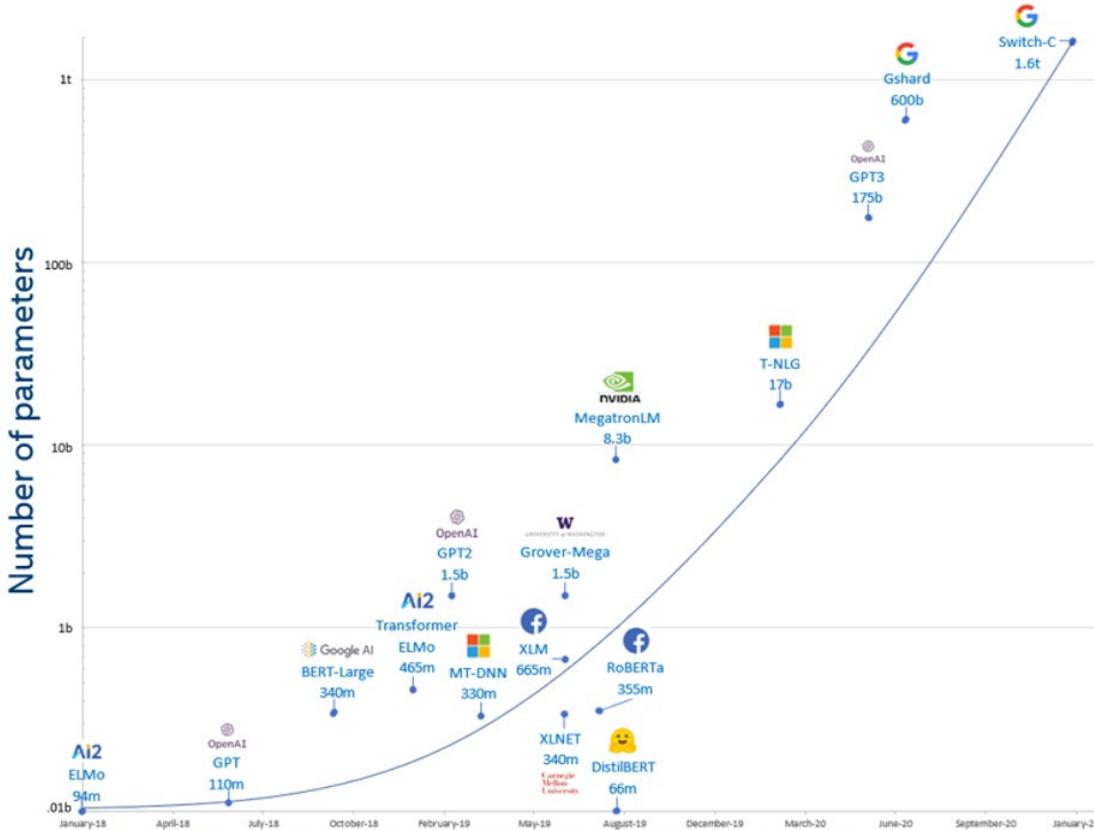
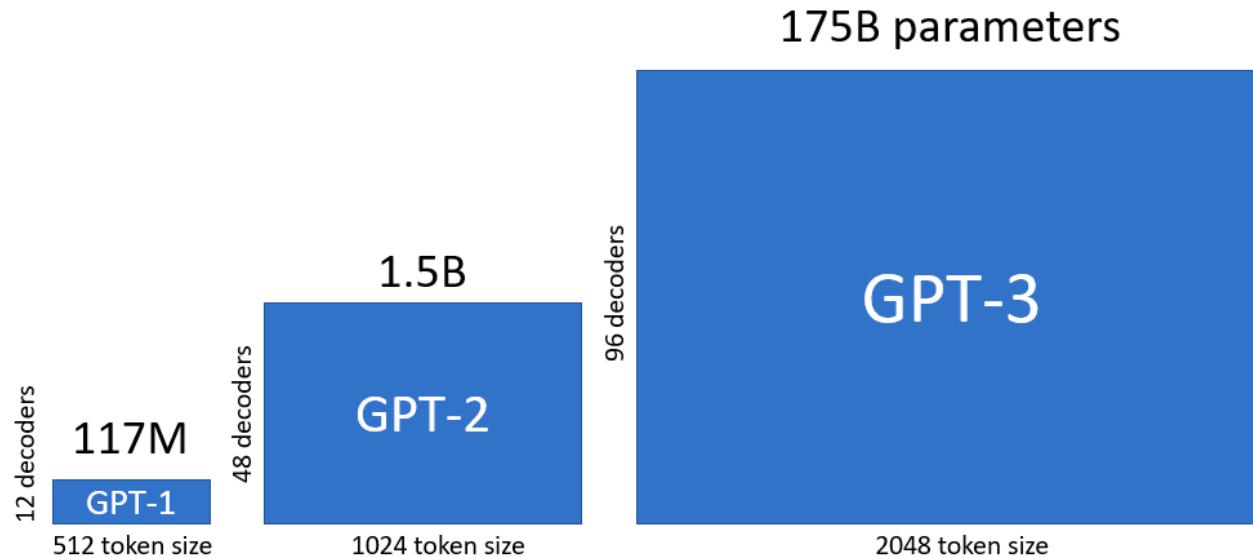
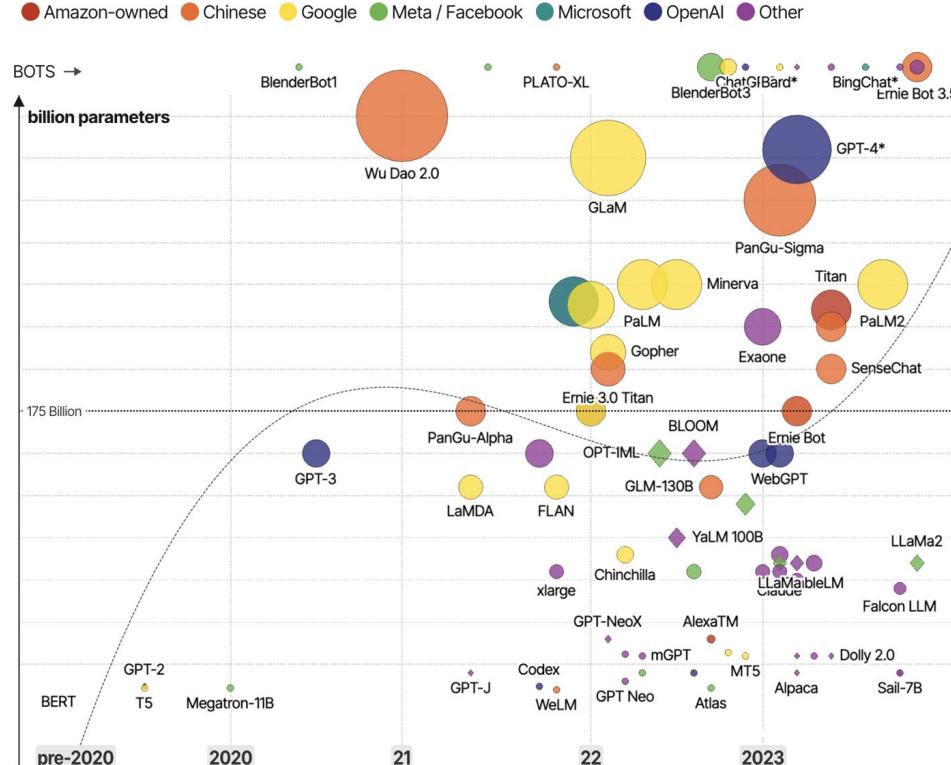


Figure 1: Exponential growth of number of parameters in DL models

Model complexity and exponential scaling



New players emerging with very large models



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

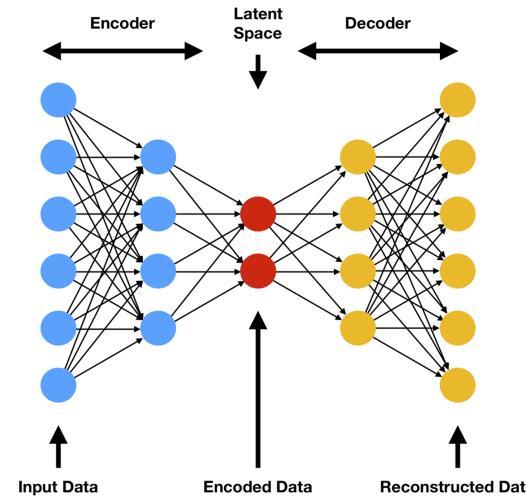
Chapter 2

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

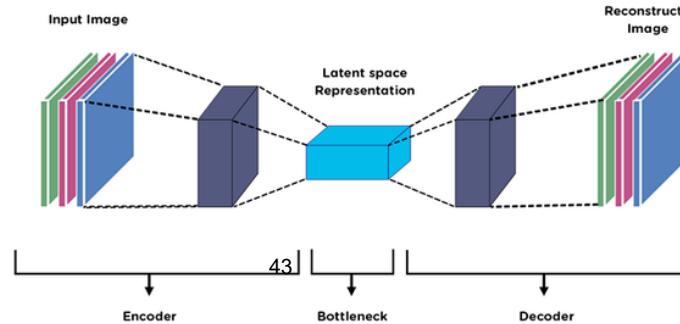


Autoencoder

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:

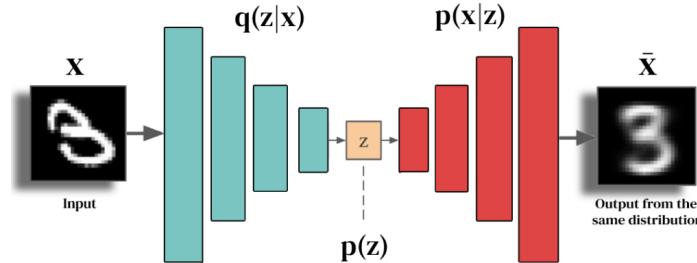
- **Encoder:**
 - The encoder takes the input data and maps it to a lower-dimensional latent space representation.
- **Decoder:**
 - The decoder takes the compressed representation from the **encoder** and reconstructs the original data from it.



Variational Inference

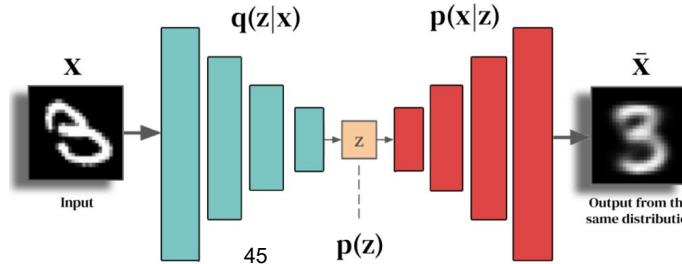
What is Variational Inference?

- **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



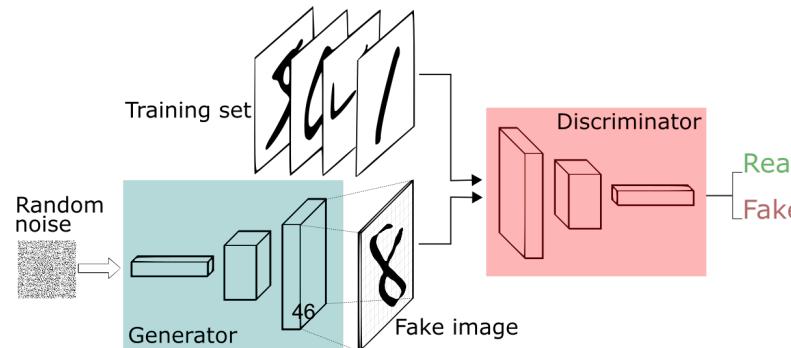
Variational Autoencoders

- 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.



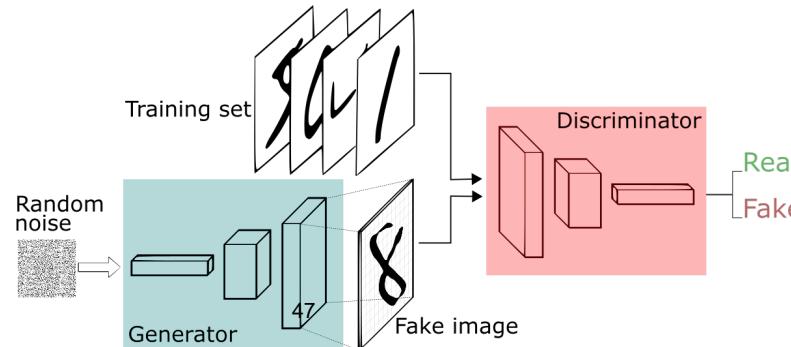
Generative Adversarial Networks (GAN)

- 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



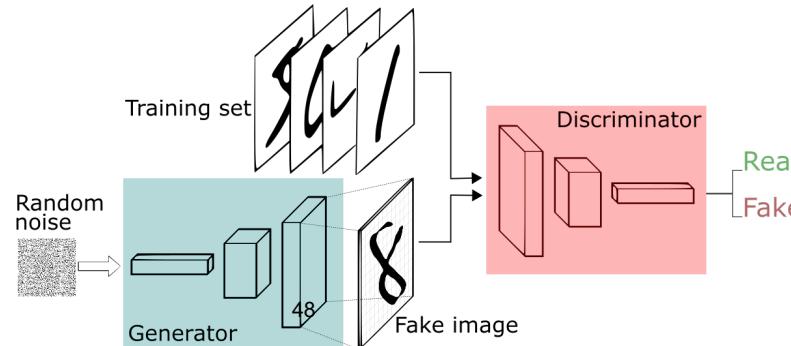
How GANs work

- 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

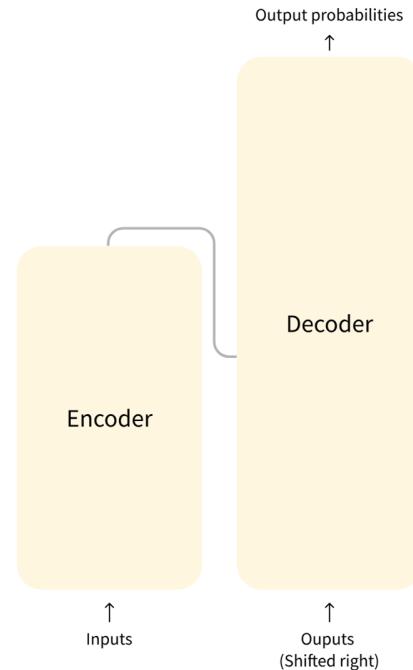
Transformers Overview

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

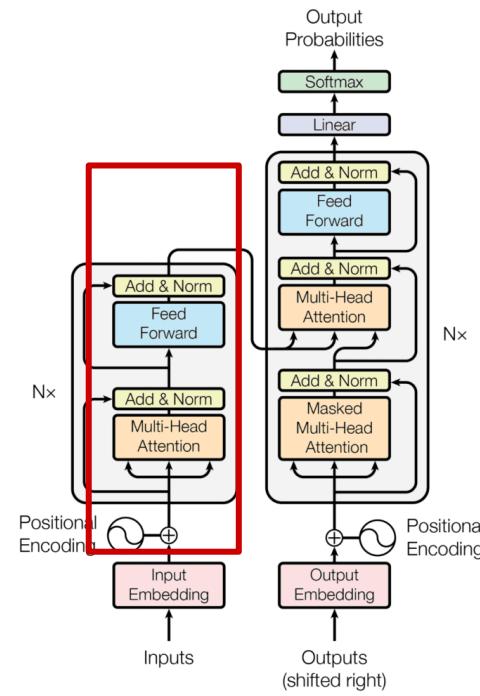
use case: predict next word(s) in a sentence or text string

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.

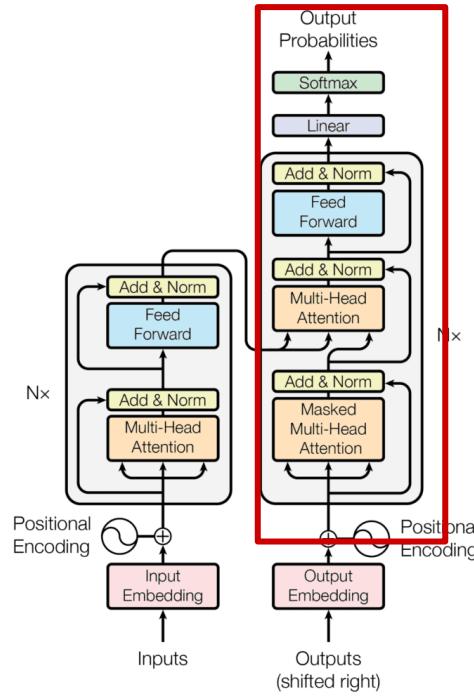
Transformer Architecture

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



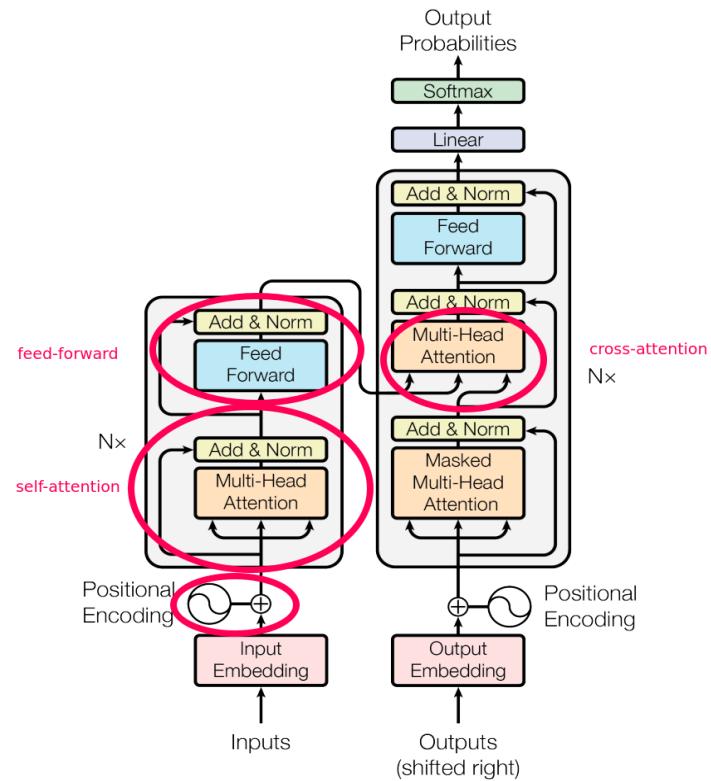
Transformer Architecture

- 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.



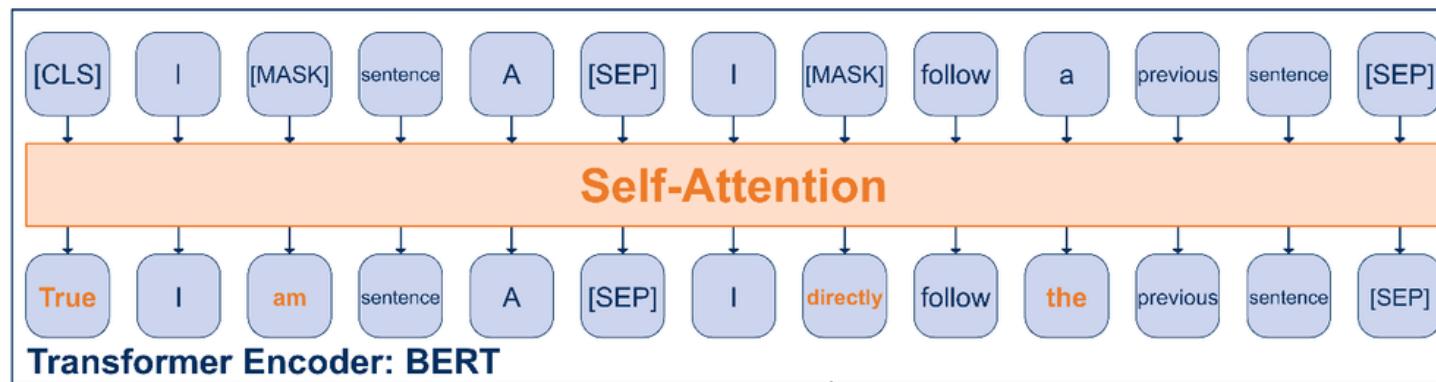
Transformer Architecture

- **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.



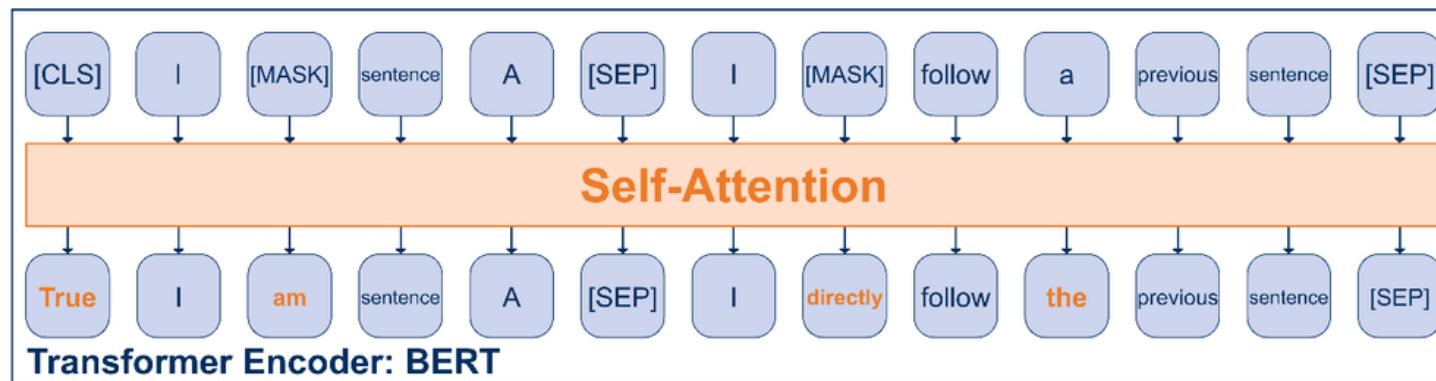
Formatting Data for Training Transformers

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



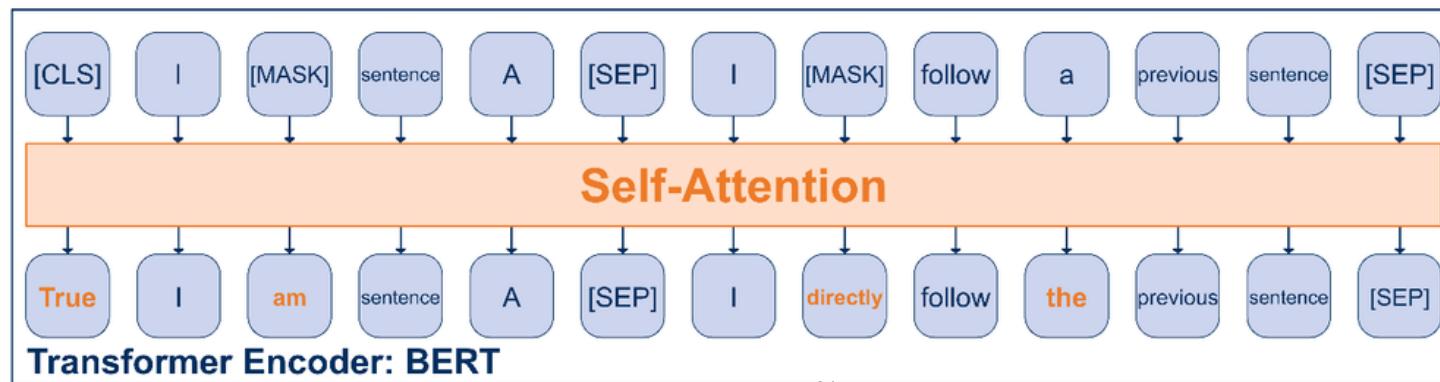
Formatting Data for Training Transformers

- [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

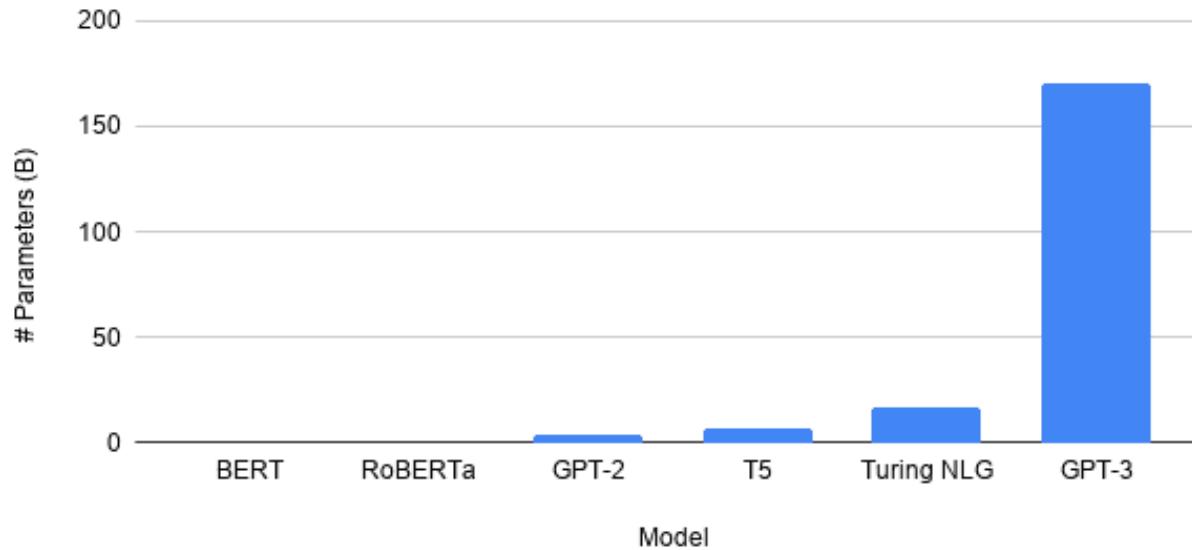
- GPT stands for **Generative Pre-trained Transformer**
- Developed by OpenAI.
- It can:
 - Understand
 - Generate
 - Manipulate human-like text
- GPT has transformed natural language processing and found applications in various fields.



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?

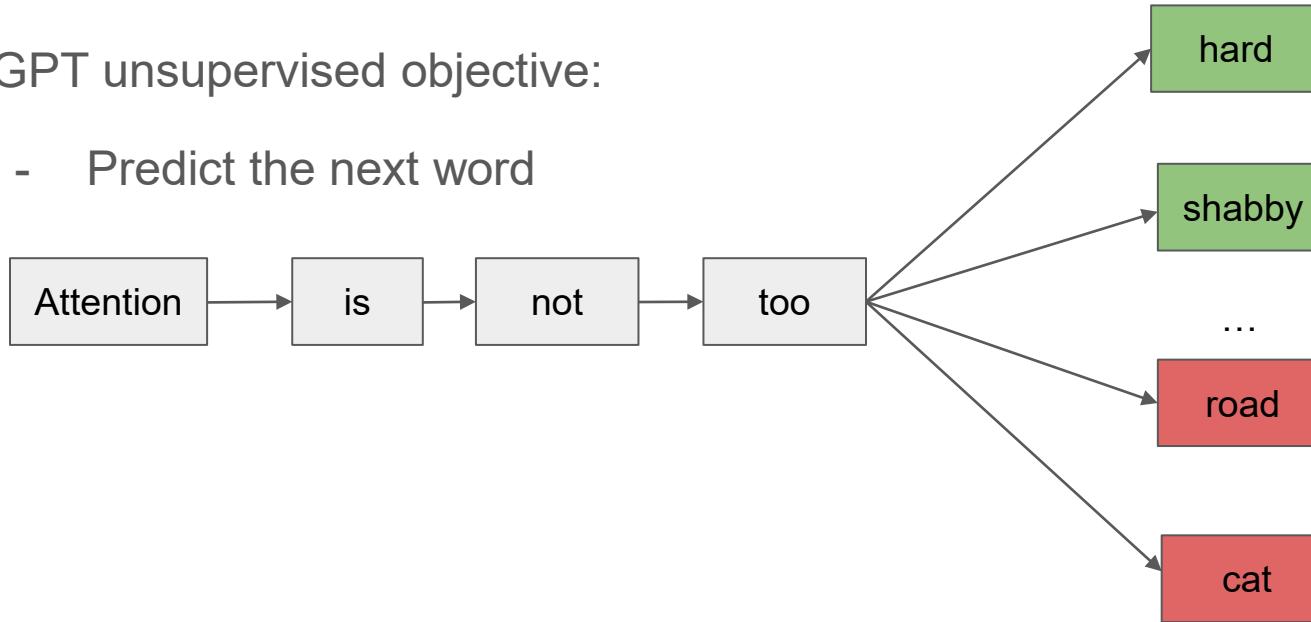
Language Modeling: Predicting the Next Token

- 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

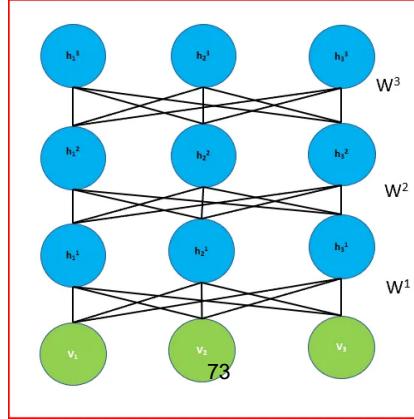
Why does it take long to update GPT model?

- **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.

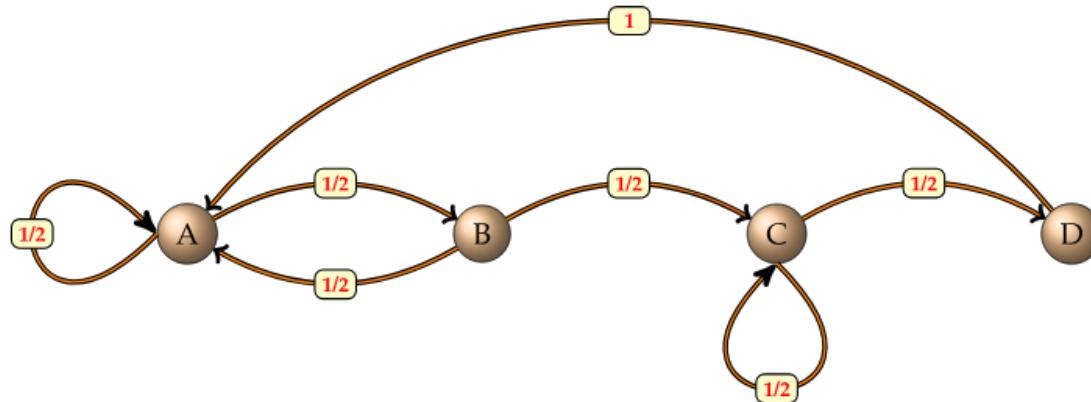
Current Status:

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

Markov Chain Monte Carlo

- **Definition**

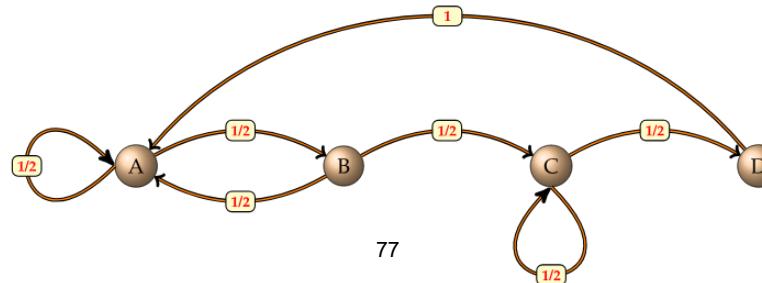
Markov Chains are a mathematical model used to represent a stochastic process, where the future state of the system depends only on the current state and not on the sequence of events that preceded it.



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.

Building Generative AI Applications (part 1)

Chapter 3

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

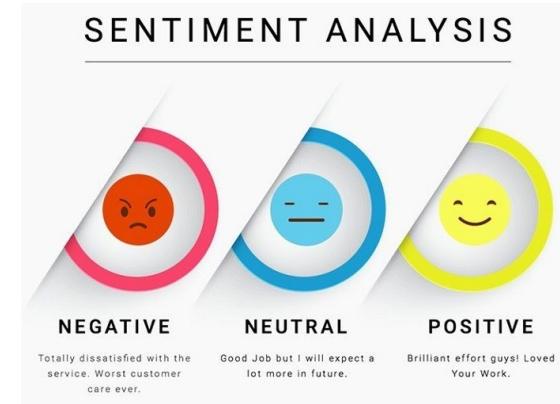
Sentiment Analysis

- 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:

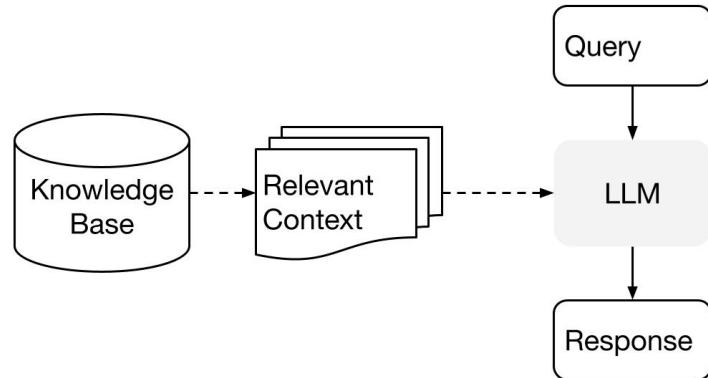
- Bias and Fairness
 - LLMs often inherit biases from the data they are trained on, which can result in biased language generation.
- Ethical Concerns
 - Generation of misinformation information, data privacy.
- Resource Intensiveness
 - Training and fine-tuning large LLMs require big computational resources.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

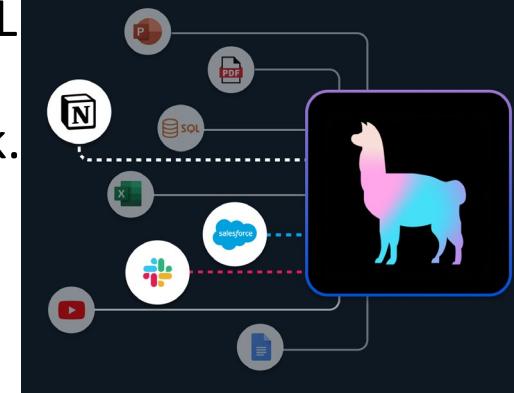
- Indexing stage
 - Preparing a knowledge base, and
- Querying stage
 - retrieving relevant context from the knowledge base



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.



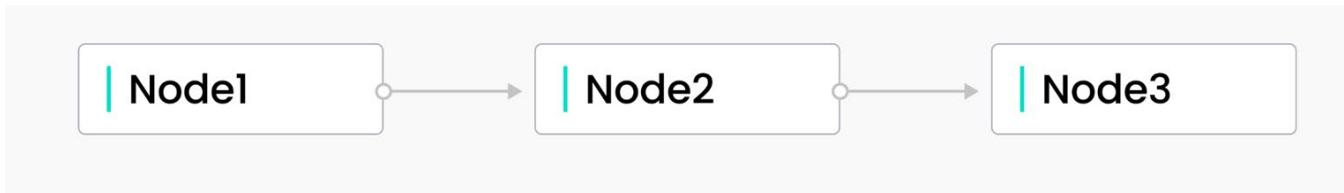
Types of Indexes in Llama Index

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.

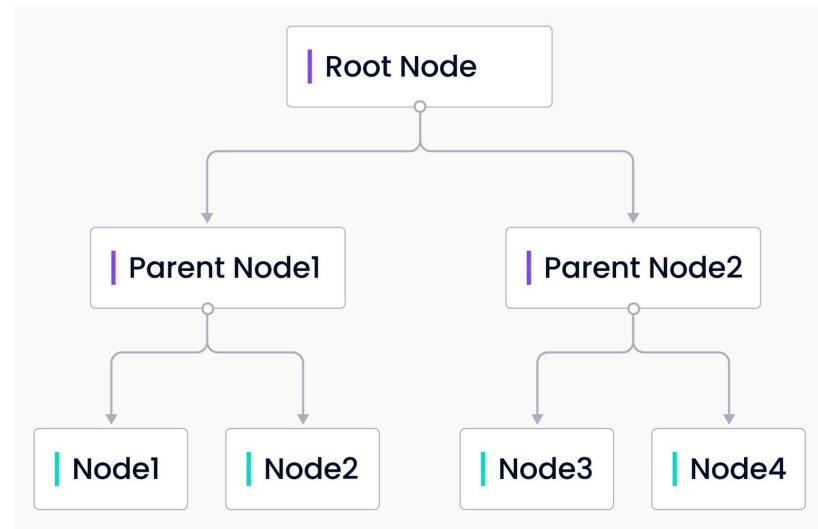


Types of Indexes in LlamaIndex - Tree Index

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:

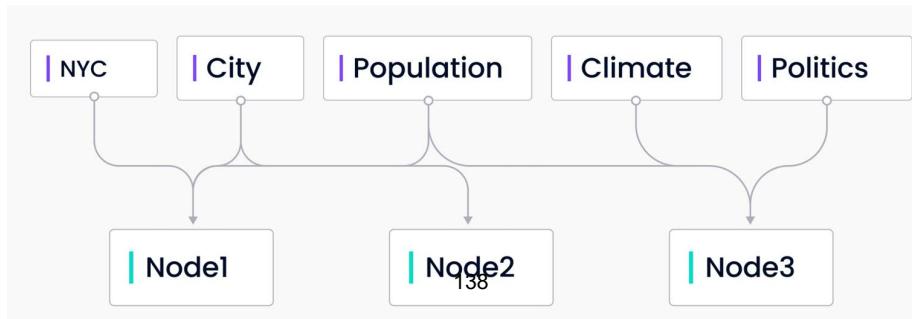
- Retrieve information from long lengthy document (from specific sections)

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	Efficient and cost-effective indexing and retrieval.	The quality of answers depends on the quality of embeddings

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

The screenshot shows the Copy AI interface. On the left is a sidebar with the 'copy.ai' logo at the top. Below it are several menu items: Chat (with a speech bubble icon), My Projects (with a folder icon), Infobase (with a database icon), Brand Voice (with a microphone icon), and Workflows (with a gear icon). A purple button labeled 'Upgrade to Pro' with a checkmark is visible. At the bottom of the sidebar, it says 'Tricia's Workspace' and 'Free Plan'. The main area has a title '2023-10-05 Untitled' and a 'New Project' button. The central content area features a heading 'Welcome to Chat by Copy.ai' with a subtext: 'Get started by writing a task and Chat can do the rest. Not sure where to start? Check out the Prompt Library for inspiration.' To the right of this text is a green vertical bar with a white pencil icon and the word 'Editor'. Below the welcome text are two sections: 'Real-Time Search' (with a magnifying glass icon) containing prompts like 'Summarize the latest news on generative AI!' and 'Write a personalized email to [insert LinkedIn profile URL]'; and 'Long Form Content' (with a document icon) containing prompts like 'Create a blog post about search engine optimization' and 'Write a press release about www.copy.ai'. At the bottom, there is a search bar with the placeholder 'Ask or search anything', a 'Browse Prompts' button, a 'No Brand Voice' button, and an 'Improve' button with a play icon.

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.

154

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, and Sales, with Email Marketing currently selected. The main area displays a preview of the workspace prompt, which is based on the 'Attention-Interest-Desire-Action' framework. The preview text asks to write an email marketing campaign that highlights features, explains advantages, and elaborates on benefits. The interface includes a search bar at the top, a preview section on the right, and a 'Use Prompt' button at the bottom right.

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 logo icon. Description: Use the oldest marketing framework in the world. Attention, Interest, Desire, Action.
- PAS Framework**: PAS logo 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 1 158 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. At the top, there is a navigation bar with the Microsoft Bing logo, a search bar labeled "SEARCH", a "CHAT" button (which is underlined, indicating it is active), and user account information including "Sign in", "15", and a profile icon. Below the navigation bar is the large blue "B" logo. The main heading "Welcome to the new Bing" is displayed prominently, followed by the subtitle "Your AI-powered copilot for the web".

Three call-to-action cards are shown below:

- 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"

At the bottom of the page, there is a feedback section with a "New topic" button, an input field for "Ask me anything...", a "Feedback" button, and a character count indicator "0/2000". A status message "Waiting for www.bing.com" is visible at the very bottom left.

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

162

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

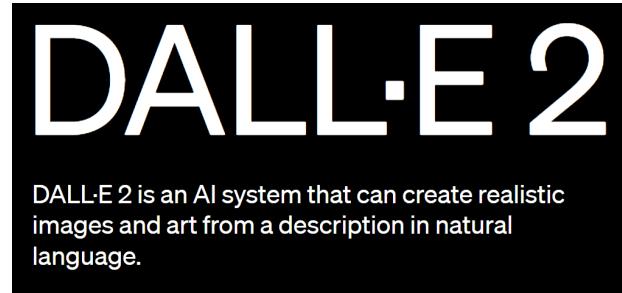
- Image Synthesis is about creating new images from scratch or transforming existing ones.
- Examples include DALL-E by OpenAI which creates images from text descriptions, and StyleGAN, a model by Nvidia capable of creating highly realistic human faces.

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 bar are three dots and a profile icon. Below the 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. Below the input field, there is a smaller text input field with the placeholder "Start with a detailed description" and a "Surprise me" button next to it. Further down, 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

DEEPART.io Pricing Latest images Create image About Register Sign in

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 these settings is a 'HIDE ADVANCED SETTINGS' link. The main area features a piano-roll style visualization of musical notes being generated, with a progress bar indicating the process. At the bottom, there is a play button labeled 'PLAY FROM START', download and tweet links, and a 'RESET' button. A note at the bottom states: "Some of MuseNet's limitations include: 176 • The instruments you ask for are strong suggestions, not requirements. MuseNet" followed by a long URL.

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

The screenshot shows the Synthesia interface. At the top, there's a toolbar with icons for Shape, Media, and Record. A speech bubble says "This is what your video will look like". Below it is a video frame showing a woman with long dark hair, Natalie, in a dark blue sleeveless dress against a blue abstract background. To the right of the video frame is a sidebar titled "Add avatar" with five AI character profiles: Natalie (woman), Jayathien (man), Laura (woman), and Kenneth (man). At the bottom of the interface, there are controls for language (English (US) - Engaging), gestures, marker, pause, and diction, along with a progress bar showing 0.00 / 0.09. The generated video script is displayed at the bottom: "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**

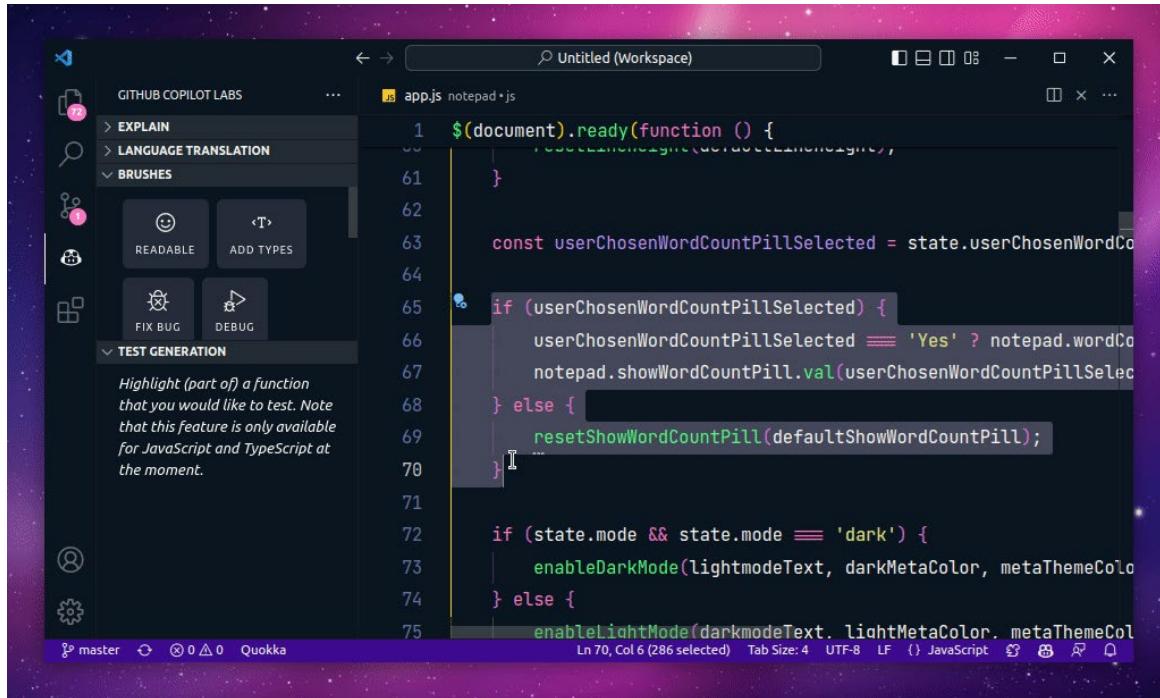
Code Generation: GitHub Copilot

- **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**



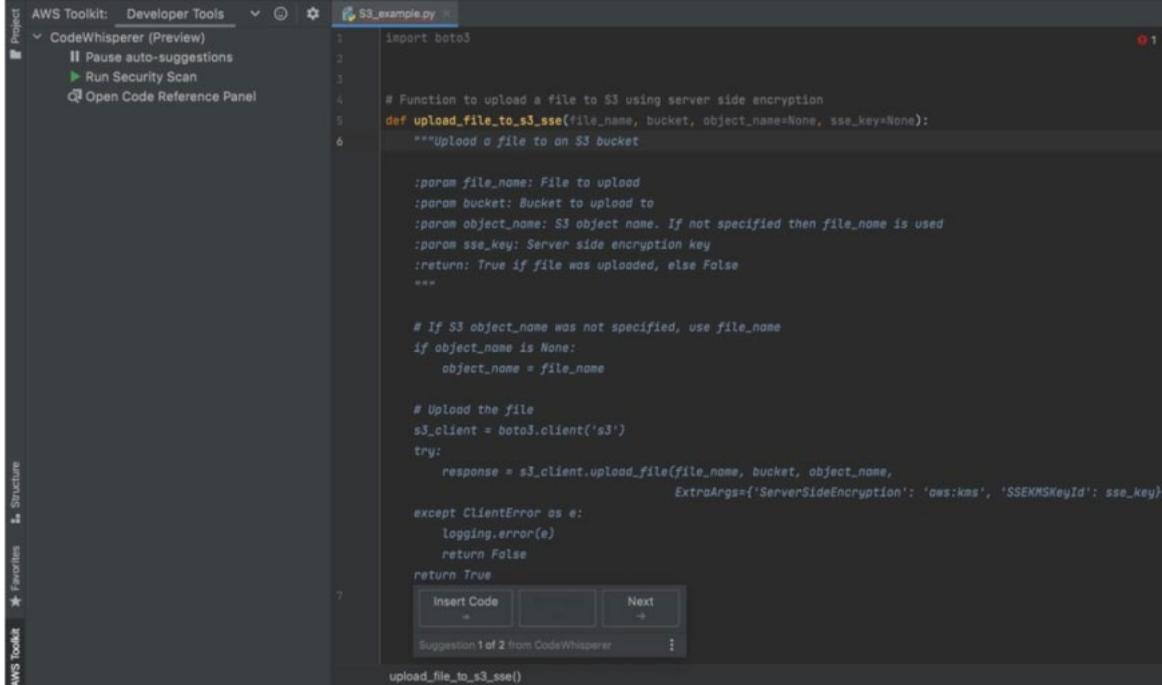
CodeWhisperer

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 displays project settings, including 'CodeWhisperer (Preview)' which has 'Pause auto-suggestions' and 'Run Security Scan' options. The main editor window contains Python code for uploading files to S3 using server-side encryption. A tooltip at the bottom indicates 'Suggestion 1 of 2 from CodeWhisperer'. The code is as follows:

```
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
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
59
60
61
62
63
64
65
66
67
68
69
69
70
71
72
73
74
75
76
77
78
79
79
80
81
82
83
84
85
86
87
88
89
89
90
91
92
93
94
95
96
97
98
99
99
100
101
102
103
104
105
106
107
108
109
109
110
111
112
113
114
115
116
117
118
119
119
120
121
122
123
124
125
126
127
127
128
129
129
130
131
132
133
134
135
136
137
138
139
139
140
141
142
143
144
145
146
147
148
149
149
150
151
152
153
154
155
156
157
158
159
159
160
161
162
163
164
165
166
167
168
169
169
170
171
172
173
174
175
176
177
178
179
179
180
181
182
183
184
185
186
187
188
189
189
190
191
192
193
194
195
196
197
198
199
199
200
201
202
203
204
205
206
207
208
209
209
210
211
212
213
214
215
216
217
217
218
219
219
220
221
222
223
224
225
226
227
227
228
229
229
230
231
232
233
234
235
236
237
237
238
239
239
240
241
242
243
244
245
246
247
247
248
249
249
250
251
252
253
254
255
256
257
257
258
259
259
260
261
262
263
264
265
266
267
267
268
269
269
270
271
272
273
274
275
276
277
277
278
279
279
280
281
282
283
284
285
286
287
287
288
289
289
290
291
292
293
294
295
296
297
297
298
299
299
300
301
302
303
304
305
306
307
307
308
309
309
310
311
312
313
314
315
315
316
317
317
318
319
319
320
321
322
323
324
325
326
326
327
328
328
329
330
331
332
333
334
335
336
336
337
338
338
339
340
341
342
343
344
345
345
346
347
347
348
349
349
350
351
352
353
354
355
356
356
357
358
358
359
360
361
362
363
364
365
365
366
367
367
368
369
369
370
371
372
373
374
375
375
376
377
377
378
379
379
380
381
382
383
384
385
385
386
387
387
388
389
389
390
391
392
393
394
395
395
396
397
397
398
399
399
400
401
402
403
404
405
405
406
407
407
408
409
409
410
411
412
413
414
415
415
416
417
417
418
419
419
420
421
422
423
424
425
425
426
427
427
428
429
429
430
431
432
433
434
435
435
436
437
437
438
439
439
440
441
442
443
444
445
445
446
447
447
448
449
449
450
451
452
453
454
455
455
456
457
457
458
459
459
460
461
462
463
464
465
465
466
467
467
468
469
469
470
471
472
473
474
475
475
476
477
477
478
479
479
480
481
482
483
484
485
485
486
487
487
488
489
489
490
491
492
493
494
495
495
496
497
497
498
499
499
500
501
502
503
504
505
505
506
507
507
508
509
509
510
511
512
513
514
515
515
516
517
517
518
519
519
520
521
522
523
524
525
525
526
527
527
528
529
529
530
531
532
533
534
535
535
536
537
537
538
539
539
540
541
542
543
544
545
545
546
547
547
548
549
549
550
551
552
553
554
555
555
556
557
557
558
559
559
560
561
562
563
564
565
565
566
567
567
568
569
569
570
571
572
573
574
575
575
576
577
577
578
579
579
580
581
582
583
584
585
585
586
587
587
588
589
589
590
591
592
593
594
595
595
596
597
597
598
599
599
600
601
602
603
604
605
605
606
607
607
608
609
609
610
611
612
613
614
615
615
616
617
617
618
619
619
620
621
622
623
624
625
625
626
627
627
628
629
629
630
631
632
633
634
635
635
636
637
637
638
639
639
640
641
642
643
644
645
645
646
647
647
648
649
649
650
651
652
653
654
655
655
656
657
657
658
659
659
660
661
662
663
664
665
665
666
667
667
668
669
669
670
671
672
673
674
675
675
676
677
677
678
679
679
680
681
682
683
684
685
685
686
687
687
688
689
689
690
691
692
693
694
695
695
696
697
697
698
699
699
700
701
702
703
704
705
705
706
707
707
708
709
709
710
711
712
713
714
715
715
716
717
717
718
719
719
720
721
722
723
724
725
725
726
727
727
728
729
729
730
731
732
733
734
735
735
736
737
737
738
739
739
740
741
742
743
744
745
745
746
747
747
748
749
749
750
751
752
753
754
755
755
756
757
757
758
759
759
760
761
762
763
764
765
765
766
767
767
768
769
769
770
771
772
773
774
775
775
776
777
777
778
779
779
780
781
782
783
784
785
785
786
787
787
788
789
789
790
791
792
793
794
795
795
796
797
797
798
799
799
800
801
802
803
804
805
805
806
807
807
808
809
809
810
811
812
813
814
815
815
816
817
817
818
819
819
820
821
822
823
824
825
825
826
827
827
828
829
829
830
831
832
833
834
835
835
836
837
837
838
839
839
840
841
842
843
844
845
845
846
847
847
848
849
849
850
851
852
853
854
855
855
856
857
857
858
859
859
860
861
862
863
864
865
865
866
867
867
868
869
869
870
871
872
873
874
875
875
876
877
877
878
879
879
880
881
882
883
884
885
885
886
887
887
888
889
889
890
891
892
893
894
895
895
896
897
897
898
899
899
900
901
902
903
904
905
905
906
907
907
908
909
909
910
911
912
913
914
915
915
916
917
917
918
919
919
920
921
922
923
924
925
925
926
927
927
928
929
929
930
931
932
933
934
935
935
936
937
937
938
939
939
940
941
942
943
944
945
945
946
947
947
948
949
949
950
951
952
953
954
955
955
956
957
957
958
959
959
960
961
962
963
964
965
965
966
967
967
968
969
969
970
971
972
973
974
975
975
976
977
977
978
979
979
980
981
982
983
984
985
985
986
987
987
988
989
989
990
991
992
993
994
995
995
996
997
997
998
999
999
1000
1000
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1610
1611
1611
1612
1612
1613
1613
1614
1614
1615
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1620
1621
1621
1622
1622
1623
1623
1624
1624
1625
1625
1626
1626
1627
1627
1628
1628
1629
1629
1630
1630
1631
1631
1632
1632
1633
1633
1634
1634
1635
1635
1636
1636
1637
1637
1638
1638
1639
1639
1640
1640
1641
1641
1642
1642
1643
1643
1644
1644
1645
1645
1646
1646
1647
1647
1648
1648
1649
1649
1650
1650
1651
1651
1652
1652
1653
1653
1654
1654
1655
1655
1656
1656
1657
1657
1658
1658
1659
1659
1660
1660
1661
1661
1662
1662
1663
1663
1664
1664
1665
1665
1666
1666
1667
1667
1668
1668
1669
1669
1670
1670
1671
1671
1672
1672
1673
1673
1674
1674
1675
1675
1676
1676
1677
1677
1678
1678
1679
1679
1680
1680
1681
1681
1682
1682
1683
1683
1684
1684
1685
1685
1686
1686
1687
1687
1688
1688
1689
1689
1690
1690
1691
1691
1692
1692
1693
1693
1694
1694
1695
1695
1696
1696
1697
1697
1698
1698
1699
1699
1700
1700
1701
1701
1702
1702
1703
1703
1704
1704
1705
1705
1706
1706
1707
1707
1708
1708
1709
1709
1710
1710
1711
1711
1712
1712
1713

```



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

198

 Open in Colab

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



Building Generative AI Applications (part 2)

Chapter 5

Agenda

- Customizing with Prompt Engineering
- Advanced Prompt Types
- Customizing with RAG
- Customizing with SYSTEM/CONTEXT Arguments and Prompt Templates
- Customizing with Fine Tuning
- Design Considerations and Tradeoffs for Customizing
- Tying It Together with LangChain

Customizing with Prompt Engineering

What is 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

Context

- **Attention Mechanism:** Both types of models often use an attention mechanism to weigh the importance of different words in the input when making predictions.
- **This is one of the most important points to remember for Prompt Engineering!**

What is Prompt Engineering?

- Definition
- Purpose

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

- **Context:** Some prompts may incorporate user or situational context for more accurate responses.
- In a chatbot, this can be given through multiple prompts
- Or, in a single prompt, specifying the context vs. the prompt instructions

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.

Advanced Prompt Types

Zero-Shot Prompting

- Simply ask a question or present a task to an LLM
- Expects the model to respond without additional context or examples

Relies on the model's pre-trained, existing knowledge

Example: Zero Shot Prompting



Classify the text into neutral, negative or positive.

Text: I think the vacation is okay.

Sentiment:

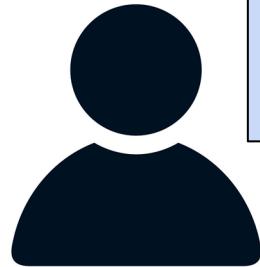
Neutral



Few-Shot Prompting

Presenting a model with a task or question along with a few examples of the desired output.

Example: Few Shot Prompting



Q: Susie had 5 Twix. She ate one, then got two more from her friend. How many does Susie have now?

A: Susie had 5, then ate one, so she has 4 left. Then she got 2 more, and now she has 6 Twix bars.

Q: Bob has 38 snickers bars. He eats 14 snickers bars, but Jim gives him 2 twix bars. How many candy bars does Bob have?

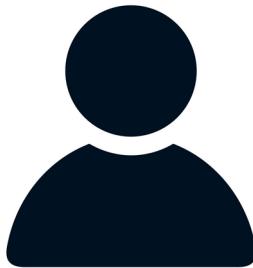
Bob had 38 snickers bars. After eating 14 of them, he has 24 snickers bars left. He then gets 2 twix bars from Jim, so he now has 26 candy bars total.



Chain-of-Thought Prompting

- Enhances complex reasoning capabilities through intermediate reasoning steps
- Can be triggered through a phrase in the prompt like, “Think through the problem”
- Can also be triggered with examples
- Some recent LLM’s will go into COT reasoning with certain zero-shot prompts on their own.

Example: CoT Prompting



Bob has 38 snickers bars. He eats 14 snickers bars, but Jim gives him 2 twix bars. How many candy bars does Bob have?

Think through the problem

Bob had 38 snickers bars and he ate 14 snickers bars. That means he had 24 snickers bars left. If Jim gave him 2 twix bars, then Bob now has 26 candy bars.



Example: Combining Multiple Techniques



Q: Susie had 5 Twix. She ate one, then got two more from her friend. How many does Susie have now? Think through the problem.

A: Susie had 5, then ate one, so she has 4 left. Then she got 2 more, and now she has 6 Twix bars.

Q: Bob has 38 snickers bars. He eats 14 snickers bars, but Jim gives him 2 twix bars. How many candy bars does Bob have? Think through the problem.

Bob starts with 38 Snickers bars. He eats 14 of them, so he is now left with 24 Snickers bars. His friend Jim gives him 2 Twix bars, so now Bob has a total of 26 candy bars.



Self Consistency Prompting

- Prompts the LLM multiple times with the same prompt
- Takes the majority answer
- Wang et al. (2022)
- Replaces the naive greedy decoding used in chain-of-thought prompting
- Often used with few-shot CoT

Generated Knowledge Prompting

- By Liu et al. 2022
- Knowledge can be integrated at inference time. Showing that reference knowledge can be used instead of model fine tuning.
- Tested across multiple datasets, common-sense reasoning, etc.

Example: Generated Knowledge Prompting - Step 1

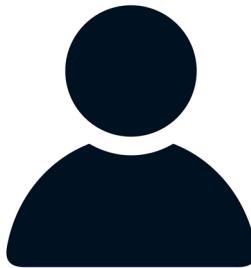


What are SEO keywords to use for a dental practice?

Dentist, Dental office, Teeth cleaning, Teeth whitening, Oral health, Gum disease



Example: Generated Knowledge Prompting - Step 2



Using the following keywords, write a blog about "Why Flossing is Important":

Dentist, Dental office, Teeth cleaning, Teeth whitening, Oral health, Gum disease

At our dental office, we often emphasize the significance of flossing during teeth cleaning appointments. Many people believe that brushing alone is enough to keep their teeth clean, but the truth is that flossing plays a vital role in maintaining oral hygiene...



Tree of Thought (ToT) Prompting

- By [Yao et al. 2023](#)
- Expands the CoT approach to explore multiple directions
- Allows the LLM to follow multiple lines of reasoning

Example: Tree of Thought Prompting



Give me 3 product ideas for an AI startup. Then give SEO keywords for each.

1. AI Virtual Assistant for Small Businesses
 - SEO keywords: AI virtual assistant, small business automation, virtual assistant software
2. AI Chatbot for Customer Service
 - SEO keywords: AI chatbot, customer service automation, chatbot software ...



Automated Prompt Engineering (APE)

- By [Zhou et al. 2022](#)
- The LLM is used to write a better prompt
- The enhanced prompt is then used to prompt the LLM

Example: Automated Prompt Engineering (APE) - Step 1



Turn this into a better prompt:
I need SEO for a marketing website

How can I effectively optimize a marketing website for search engines?



Example: Automated Prompt Engineering (APE) - Step 2



How can I effectively optimize a marketing website for search engines?

1. Conduct Keyword Research: ...
2. Optimize On-Page Elements: ...
3. Create High-Quality Content: ...



Retrieval Augmented Generation (RAG)

- Additional content is retrieved about a given question
- This is added to the prompt
- The generated response will take that additional information into account
- This is an extremely common technique in applications
- This can outperform fine tuning an LLM

Example: Retrieval Augmented Generation (RAG)



Does Denis prefer dogs or cats?

He wrote this article about dogs:

Dogs are loyal companions: Dogs are known as “man’s best friend” for a reason. They form strong bonds with their owners and are always there to offer comfort and support.

Based on this article, Denis would prefer dogs.



Automatic Reasoning and Tool-use (ART)

- Combining CoT prompting and tools integration
- The LLM creates a sequence of actions
- Then it drives external tools to execute

ReAct Prompting

- REason + ACT
- The LLM is used to generate a set of reasoning steps
- Like ART, these are then integrated with external tool use
- However, in REACT, the LLM reevaluates the next step based on the results of the previous step
- In complex sequences, there is often a loop back to the beginning, in case the desired tool interaction fails
- This is popular with LLM Agents

Customizing with RAG

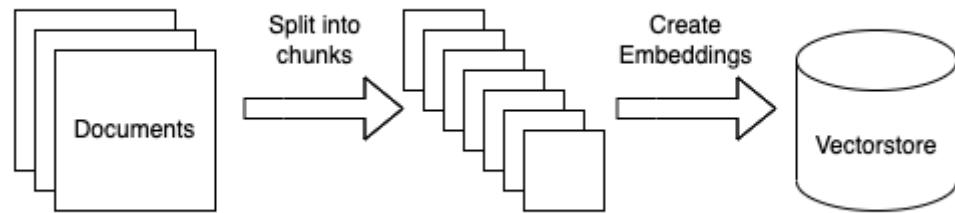
What is RAG?

1. RAG stands for Retriever-Augmented Generation.
2. It's a two-step NLP framework that combines the capabilities of retrievers and generators.
3. It enhances the model's performance by adding context to the user prompt

How Does the Retriever Work?

1. The retriever scans a large dataset or corpus to find relevant text passages.
2. Uses algorithms like BM25 or neural networks to find the most relevant information.
3. Outputs these passages for the next step.

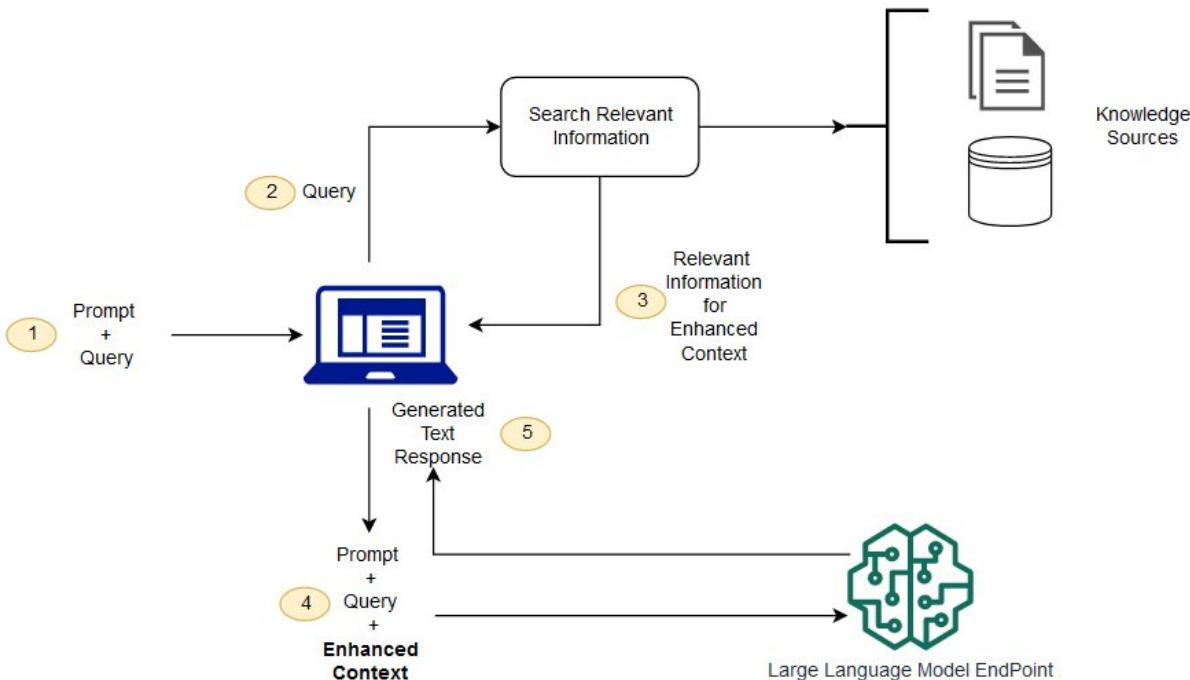
Vector Indexing



How Does the Generator Work?

1. Takes the passages retrieved by the retriever.
2. Uses these as context for generating a detailed and coherent response.
3. This content can be put in as a SYSTEM or CONTEXT argument (for some LLM's), or in a Prompt Template

RAG Architecture



Customizing with System Prompts and Prompt Templates

System Arguments

- Meta-information that guides the model's behavior.
- Example: {"verbosity": "low"} for short answers.
- Provides fine-tuned control over the model's output format and content

Context Arguments

- Provide extra content to supplement the user prompt
- Content from RAG may be added here
- Used in Content-Aware Chatbots

Prompt Templates

- Defined as structured queries with placeholders.
- Example: "Provide a brief summary of {Topic} focusing on its key features."
- Can be combined with reference content from RAG to create a 1-shot learning framework

Context vs System Arguments vs Prompt Templates

- Some LLM's provide a System argument, some a Context argument, some provide both
- Functionally, Context arguments, System arguments, and Prompt Templates can all be used to interject additional content and create a 1-shot (or few-shot) prompt
- All of these techniques of adding context, plus the user prompt, are still limited by the token size limit of the LLM

Context Aware Chatbots

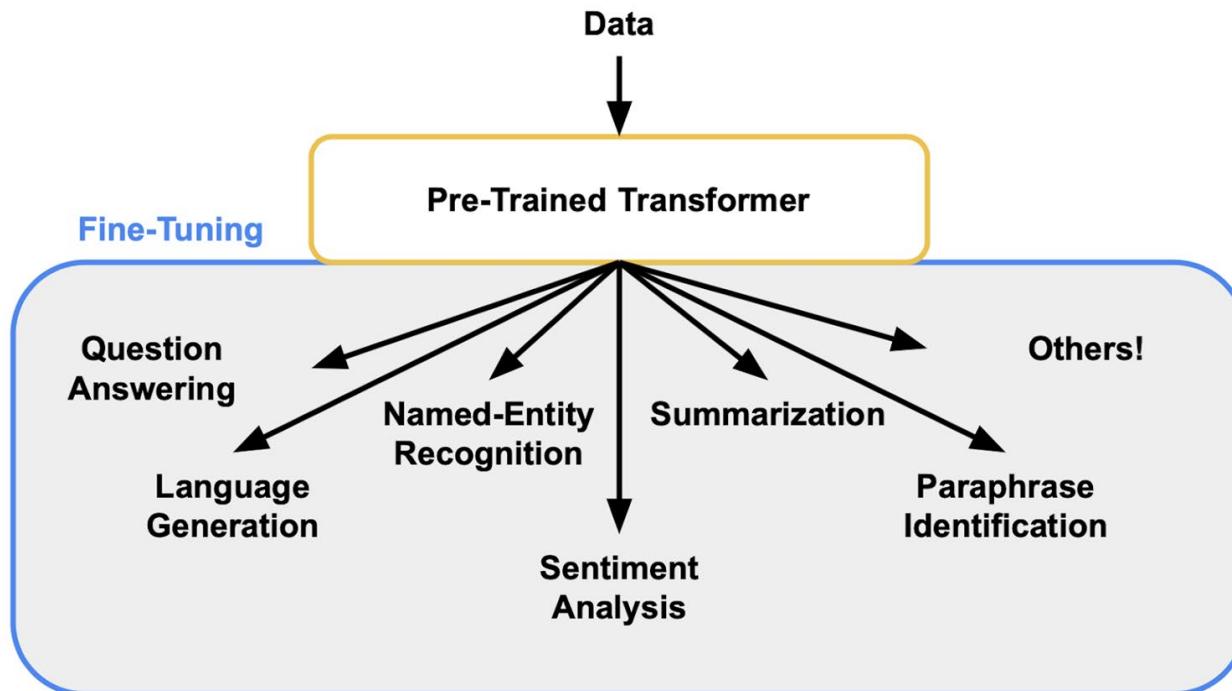
- Chatbots need to be aware of the previous conversation, not just the latest message
- The conversation history is added to the LLM query via a context argument (or prompt template, if the LLM doesn't support context arguments)
- The next response from the LLM will capture the context
- Side Note: Few-Shot prompting is actually just One-Shot prompting with the context built up over several user queries

Context Aware Chatbots

- Defined as the sequence of previous interactions between the user and the chatbot.
- Role: Provides context to interpret user queries and generate coherent responses.
- Example:
 - a. User: "Tell me a joke."
 - b. Bot: "Why did the chicken cross the road?"
 - c. User: "I don't know, why?"
 - d. Bot needs history to know that "why" refers to the joke.

Customizing with Fine Tuning

Customizing with Fine Tuning



Design Considerations and Tradeoffs

When to Fine Tune the LLM

- When you need the model to understand a specific domain language or jargon.
- If the model must adhere to strict compliance or guidelines that are difficult to enforce using only prompts and context.
- When you want to significantly alter the model's base behavior or improve its performance on a very specific type of task.

When to Use Prompt Engineering

- Good for general-purpose tasks and flexible applications.
- Suitable for applications that require quick modifications without the need for extensive re-training.
- Use when fine-grained control on a per-query basis is needed.

Trade-Offs

- Prompt Engineering and Fine Tuning are not mutually exclusive, and are often used together
- Fine-tuning is resource-intensive and may require a labeled dataset.
- Prompts and context are easier to implement but may be less effective for specialized tasks.
- Often, a combination of both methods yields the best results for complex, real-world applications.

Tying It Together with LangChain

Introduction

- ChatGPT does not have access to your private data
- ChatGPT is trained until specific point in time

J What LangChain is?

[AI Response]
As of my last knowledge update in September 2021, I'm not aware of any technology or concept called "LangChain." It's possible that LangChain is a term, project, or concept that has emerged after my last update, or it could be a niche or specialized topic that I don't have information about.

If LangChain has gained prominence or recognition after September 2021, I recommend searching for the term on the internet or checking more recent sources for accurate and up-to-date information. Is there anything else I can assist you with?

J How many pdfs files do I have in my Google Drive?

[AI Response]
I'm sorry, but I don't have access to personal data about individuals unless it has been shared with me in the course of our conversation. I am designed to respect user privacy and confidentiality. Therefore, I don't know how many PDF files you have in your Google Drive or any other specific information about your personal files or accounts.

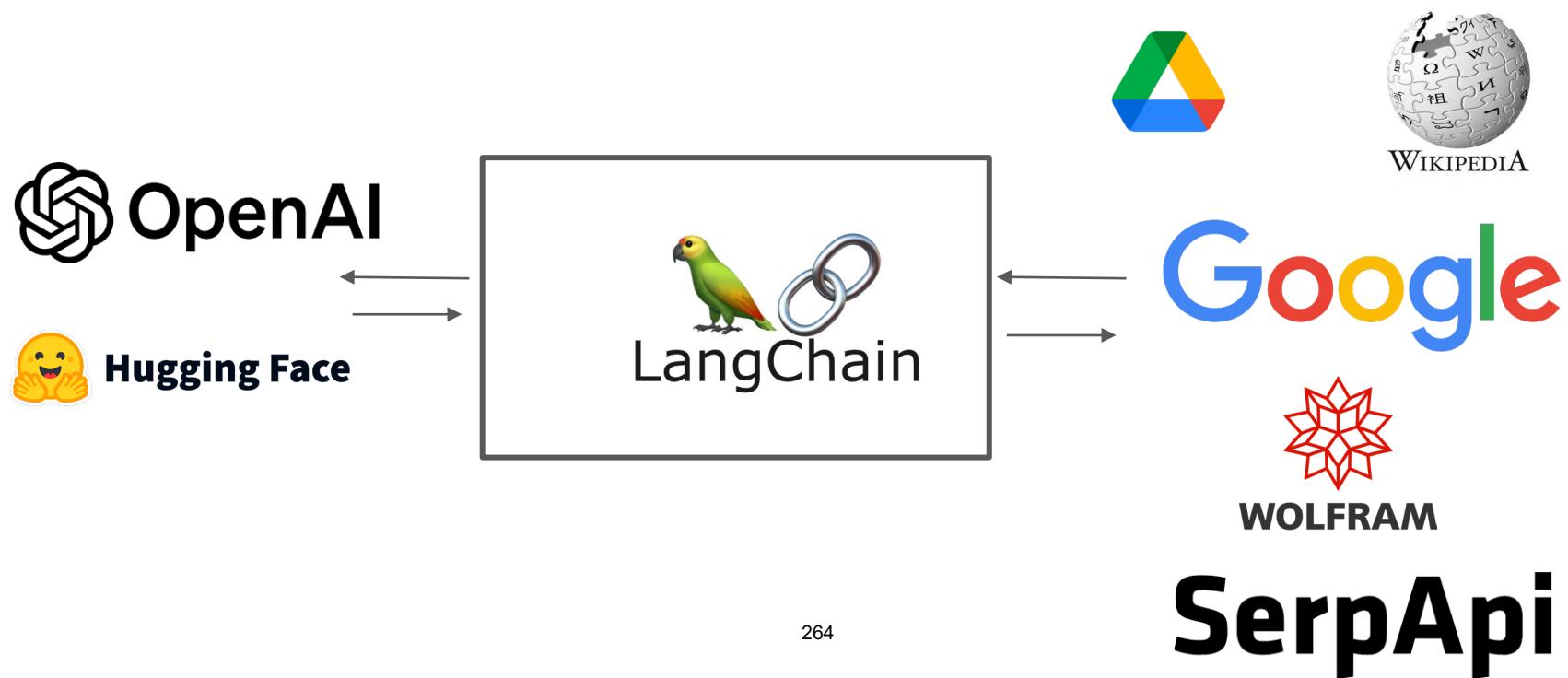
If you're looking to find out how many PDF files you have in your Google Drive, you can log in to your Google Drive account and use the search functionality to filter files by type, searching for ".pdf" to see all PDF files in your Drive.

Regenerate

Send a message ➤

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. [ChatGPT August 3 Version](#)

LangChain



What is LangChain?

- LangChain is a framework for developing applications powered by language models.
- It connects AI models with various data sources
- It is dynamic, data-responsive application with latest NLP advancements
- Ideal for developers, data scientists and NLP enthusiasts



What is LangChain?

LangChain can help you build applications in two ways:

- 1. Integration:** you can bring external data (including your files), other applications, API data into your LLMs.
- 2. Agency** - Enable your LLMs to engage with its environment through decision-making, leveraging LLMs to assist in determining the next course of action.

Why LangChain?

There are four main reasons why to use LangChain:

- 1. Components** - It contains abstractions and components necessary to work with LLMs
- 2. Customized Chains** - It allows you to combine series of actions
- 3. Speed**
- 4. Community** - Great community of users

Core Components of LangChain

The followings are the core components of LangChain:

- 1. Model I/O**
- 2. Retrieval**
- 3. Chains**
- 4. Memory**
- 5. Agents**

Core Components of LangChain - Model I/O

Module “**Model I/O**” contains the following parts:

- 1. Prompts**
- 2. Language models**
- 3. Output Parsers**

Core Components of LangChain - Model I/O

1. Prompts

There are many **prompt templates** and **Example Selectors** available, check [[LINK](#)]

Example of **Prompt Template**:

```
from langchain import PromptTemplate

prompt_template = PromptTemplate.from_template(
    "Tell me a {adjective} joke about {content}."
)
prompt_template.format(adjective="funny", content="chickens")
```

Core Components of LangChain - Model I/O

2. Language models

LangChain provides interfaces and integrations for two types of models:

- **LLMs**: Models that take a text string as input and return a text string
- **Chat models**: Models that are backed by a language model but take a list of Chat Messages as input and return a Chat Message

Core Components of LangChain - Model I/O

3. Output Parsers

- Language Models output text
- Do you want more structured information?
 - That is where **parsers** comes in
- You can format your output into:
 - List
 - Datetime
 - Enum
 - ... Other [[LINK](#)]

Core Components of LangChain - Retrieval

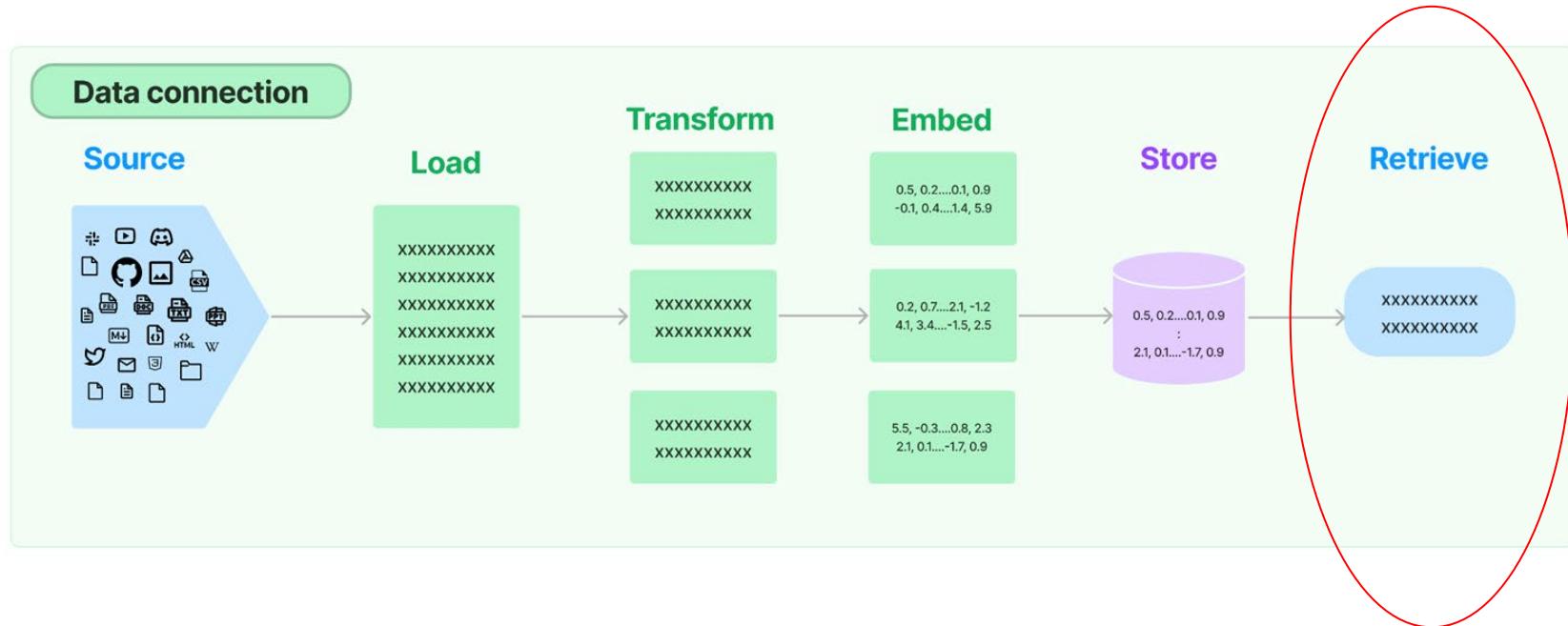
User-Specific Data Needs:

- Many LLM applications **require addition data**
- It is addressed through **Retrieval Augmented Generation (RAG)**
- External data is retrieved and used during the generation process

LangChain's Role:

- LangChain offers building blocks for RAG apps
- Despite its apparent simplicity, data retrieval can involve complexities.

Core Components of LangChain - Retrieval



Core Components of LangChain - Retrieval

Some retrieval modules in LangChain are:

- 1. Document loaders**
 - a. Loads data from documents (.txt, csv, html, pdf, YouTube transcripts etc.)
- 2. Document transformers**
 - a. Splitting long text into meaningful chunks
- 3. Text Embedding models**
 - a. Getting embedding of your input
- 4. Vector stores**
 - a. Storing embedded data and performing vector search for you
- 5. Retrievers**
 - a. Retrieving documents/vectors
- 6. Indexing**
 - a. Loading and keeping in sync documents from any source into a vector store

Core Components of LangChain - Chains

What is a chain in LangChain?

- A **chain** is a sequence of calls to components.
- Composing **components** together in a **chain** is simple but powerful.

Why do we need **Chains**?

- **Chains** allow us to create a single coherent application

Example:

- Summary #1, Summary #2, Summary #3 > **Final Summary**

Core Components of LangChain - Memory

Introduction to **Memory** in Conversational Systems

- Conversational interfaces are crucial in many LLM applications.
- **Essential element:** referencing past information within a conversation.
- **Basic requirement:** accessing a window of recent messages.

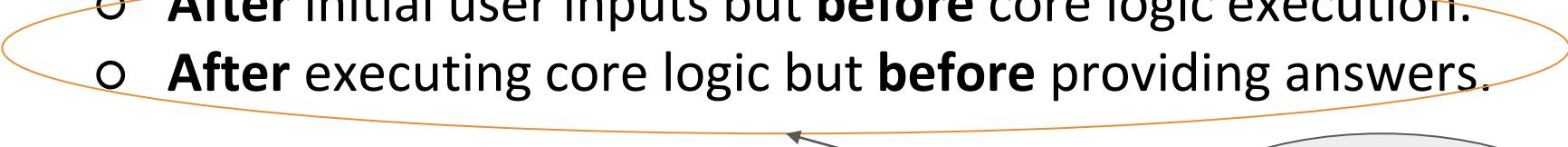
```
chat(  
  [  
    SystemMessage(content="You are a nice AI bot that helps a user figure out where to travel in one short se  
    HumanMessage(content="I like the beaches where should I go?"),  
    AIMessage(content="You should go to Nice, France"),  
    HumanMessage(content="What else should I do when I'm there?")  
  ]  
)  
  
AIMessage(content='While in Nice, you can also explore the charming Old Town and enjoy some delic:  
an cuisine.', additional_kwargs={}, example=False)
```

We did not mention
“Nice” in the last
question, it referred
from history
(memory)

Core Components of LangChain - Memory

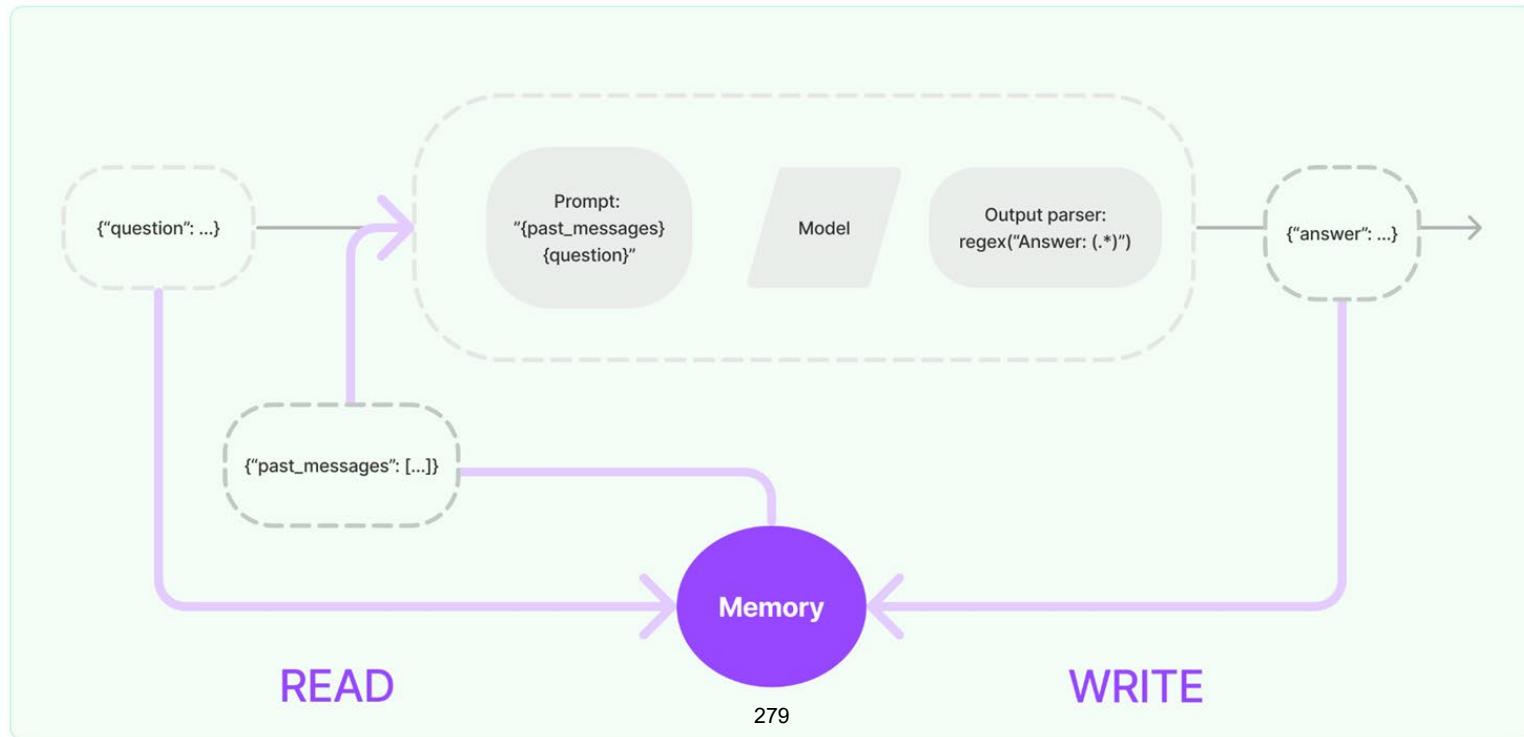
Core Actions of a Memory System

- Memory systems support two key actions: **reading** and **writing**
- Inputs come from both users and memory.
- Interaction with memory happens twice in each run:
 - **After** initial user inputs but **before** core logic execution.
 - **After** executing core logic but **before** providing answers.



Next slide - graph explanation for reading/writing

Core Components of LangChain - Memory



Core Components of LangChain - Agents

Introduction to Agents:

- **Core concept:** Employing LLMs to sequence actions
- Agents facilitate interaction with external tools
- In agents, a language model is used as a **reasoning engine** to determine:
 - **Which actions to take**
 - **What order to take actions**

Core Components of LangChain - Agents

Key Components of Agents - Agent Class:

- Responsible for determining the next step
- Utilizes language model and prompt
- Prompt includes personality, context, and strategies
 - The most common strategy is **ReAct** (Reason + Act)
 - **ReAct** selects appropriate tools based on query

Core Components of LangChain - Agents

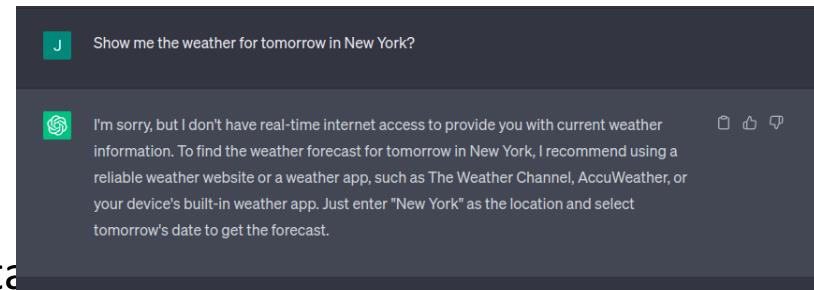
Customization and Types of Agents:

- LangChain offers various agent types.
- Customization involves adapting agents with personalized elements.
- Types of Agents:
 - Zero-Shot ReAct
 - Structured input ReAct
 - Self ask with search
 - ReAct document store
 - ...

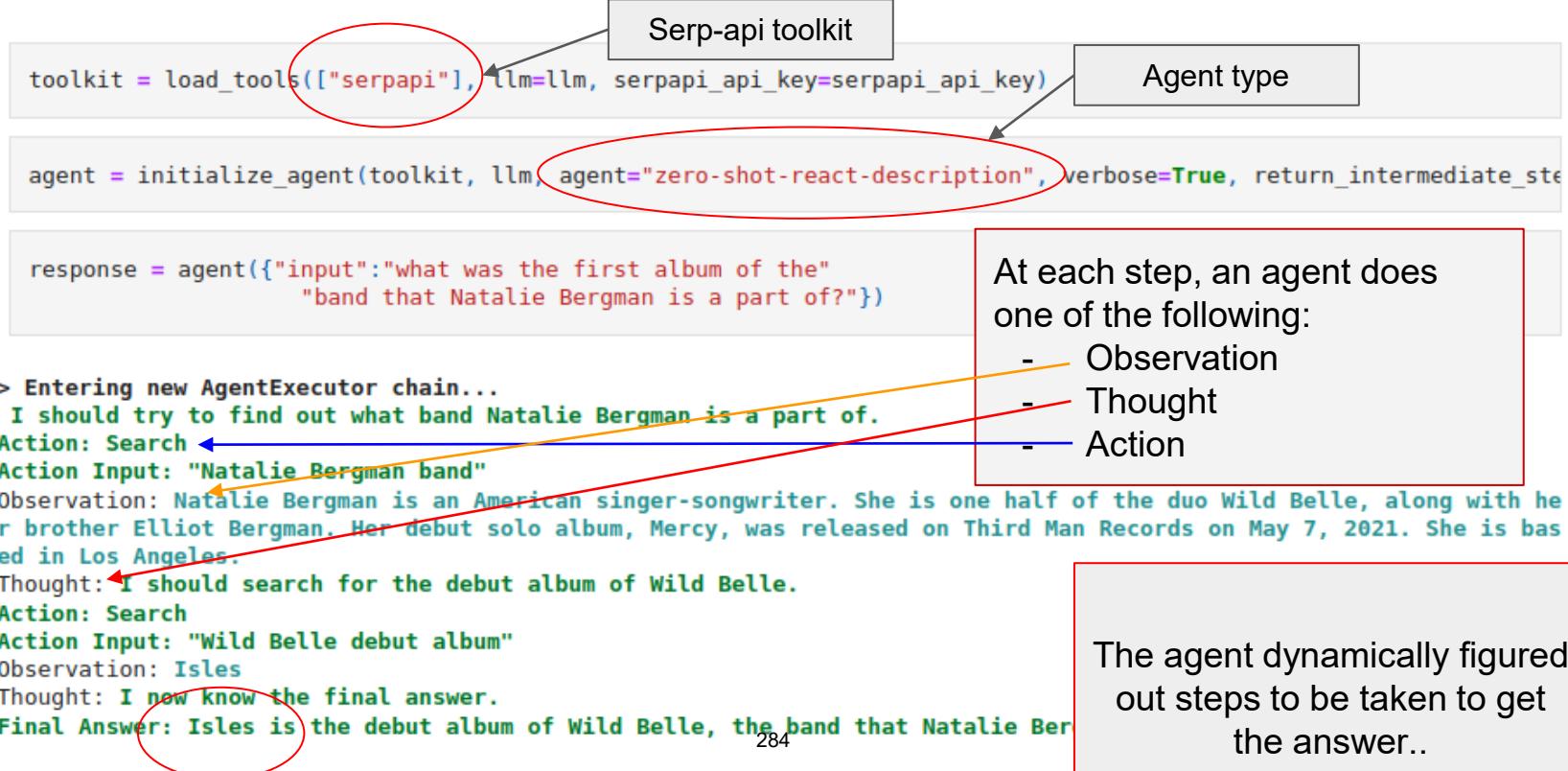
Core Components of LangChain - Agents

Example

- Asking ChatGPT about weather
 - It does **not** know
 - It does not have access to real-time data
- Imagine:
 - You send prompt (query) to an LLM in natural language (about weather)
 - Then the agent understands the query and calls weather API to fetch the data
 - This is what agents actually do
- An agent has access to specific tools
 - Google Search, Python REPL, math calculator, podcast-api, serp-api, weather APIs etc.

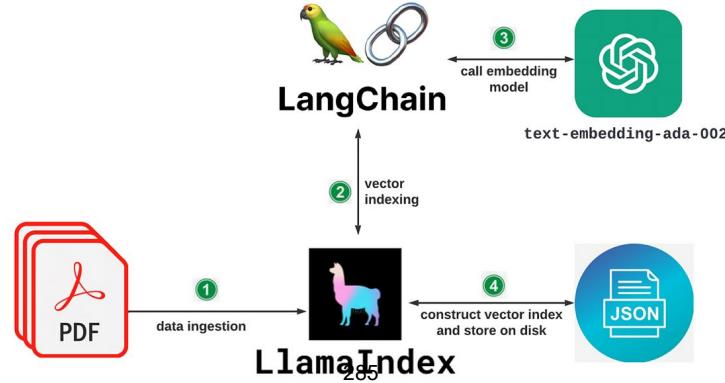


Core Components of LangChain - Agents



Llamaindex vs. LangChain

- Llamaindex focuses on indexing and efficient information retrieval (RAG) for LLMs.
- LangChain offers a broader range of LLM applications.
- Llamaindex can be used as a retrieval/tool within LangChain.



DELETED/BACKUP

TEMPLATE



Classify the text into neutral, negative or positive. Text: I think the vacation is okay.
Sentiment:

Neutral



Template BACKWARDS

Classify the text into neutral, negative or positive.Text: I think the vacation is okay.

Sentiment:



Neutral

How is Standard Prompting different than Chain-of-Thought Prompting:

Standard Prompting

- You provide a single, well-defined question or prompt to the language model and wait for its response.
- Advantages: It's straightforward and often sufficient for many types of questions. It's also more predictable.
- Limitations: The response is based entirely on that one prompt. If the prompt is vague, the answer might not be as precise as desired. Also, for complex questions, the model might not provide a complete answer in one go.

Chain-of-Thought Prompting

- This involves breaking down a problem or question into a series of smaller prompts, each building on the previous one.
- This method is used when you have a complex topic or when you want the model to think more deeply or creatively about a question.
- Advantages: Allows for deeper exploration of topics, can lead to more detailed and nuanced responses, and enables the user to guide the model more effectively.
- Limitations: It's more time-consuming and requires a better understanding of how to guide the model.

How does COT work better than N-Shot

(a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The answer is 8. X

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are $16 / 2 = 8$ golf balls. Half of the golf balls are blue. So there are $8 / 2 = 4$ blue golf balls. The answer is 4. ✓

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: The answer (arabic numerals) is

(Output) 8 X

(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: **Let's think step by step.**

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓