
AI+
NEW LIVE TRAINING
AI+
TRAINING
Hands-on
Instruction
in Practical Skills
ON-DEMAND ACCESS
INTRODUCTION
TO AI AGENTS
Sheamus McGovern
Founder and Engineer
ODSC

Upcoming ODSC Events


ODSC EUROPE
SEPTEMBER 5-6 | LONDON
ODSC.COM/EUROPE
HANDS-ON, EXPERT-LED INSTRUCTION ON THE LATEST IN AI
LEARN MORE ABOUT THE CONFERENCE

ODSC WEST 2024

AT THE HEART OF SILICON VALLEY!

OCTOBER 29-31

ODSC.COM/CALIFORNIA

JOIN US AT THE LEADING AI & DATA SCIENCE CONFERENCE


AI+
NEW LIVE TRAINING
AI+
TRAINING
Hands-on
Instruction
in Practical Skills
ON-DEMAND ACCESS
INTRODUCTION
TO AI AGENTS
Sheamus McGovern
Founder and Engineer
ODSC

Introduction to AI Agents

E2B

E2B.DEV
@E2B_DEV

AI Agents Landscape

By E2B.dev - Cloud Runtime for AI Agents

E2B users or integrations

INTEGRATED WITH E2B

Open source

Coding

Open Interpreter (E2B INTEGRATION)
Maige (RUNNING ON E2B)
Sweep AI
WorkGPT
Vanna.AI
DemoGPT
AutoPR
Aide
Smol Developer
bloop.
Automata
Continue
GPT Migrate
GPT Engineer
CodeFuse
Stackwise
Sourcegraph Cody AI
cody
ReactAgent
GPT Pilot
English Compiler

Productivity + Daily Life

Local GPT
Allice
PromethAI
Agent4Rec

Source: Awesome List of Agents on GitHub

Closed Source

Coding

Tusk
BitBuilder
v0 by Vercel
</ autopilot >
phind
Airplane Autopilot
Factory
Deepnote AI
Copilot X
Hex Magic
codium
GitLab Duo
GitWit (REACTEVAL RUNNING ON E2B)
MakeDraft
Dosu
CodeWP
grit
Input
Kusho
SECOND
mutable.ai
Butternut AI
Cursor
Codegen
Duckie AI
DevGPT

Productivity + Daily Life

Moone
MultiOn
Lindy
Spell
Claros AI Shopper
iMean.AI
AgentScale
Cykel
FLODE
Otherside Personal Assistant
Wispy
ollie
COGNOSYS
Raycast

AI Agents

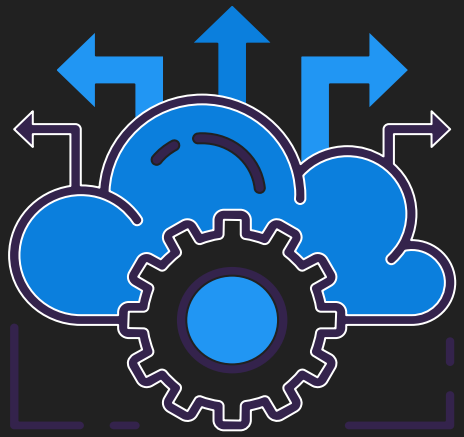
The main idea of an AI agents is to use a language model to dynamically select a sequence of actions.

Agents use a language model as a reasoning engine to decide which actions to take and in what order. This approach allows agents to be more flexible and adaptive, responding to inputs and situations in a more intelligent and context-aware manner. Agents also select which tools to execute those actions.

Traditional Agents

- Observations
- Experience

ENVIRONMENT



Actions

- Memory
- Knowledge
- Capabilities
- Goals

- Sensors

AI Agents Characteristics

You can consider a system an AI agent when it has the following characteristics:

Natural Language Understanding (NLU): An AI agent can understand and process human language in a way that is meaningful. This includes understanding syntax, semantics, and the context of the conversation.

Natural Language Generation (NLG): It can generate human-like text based on the input it receives.

Learning and Adaptation: AI agents can learn from new data and adapt their responses over time. They can be fine-tuned for specific tasks using additional training data.

Task Automation:They can perform and automate specific tasks based on user commands, such as scheduling appointments or retrieving information.

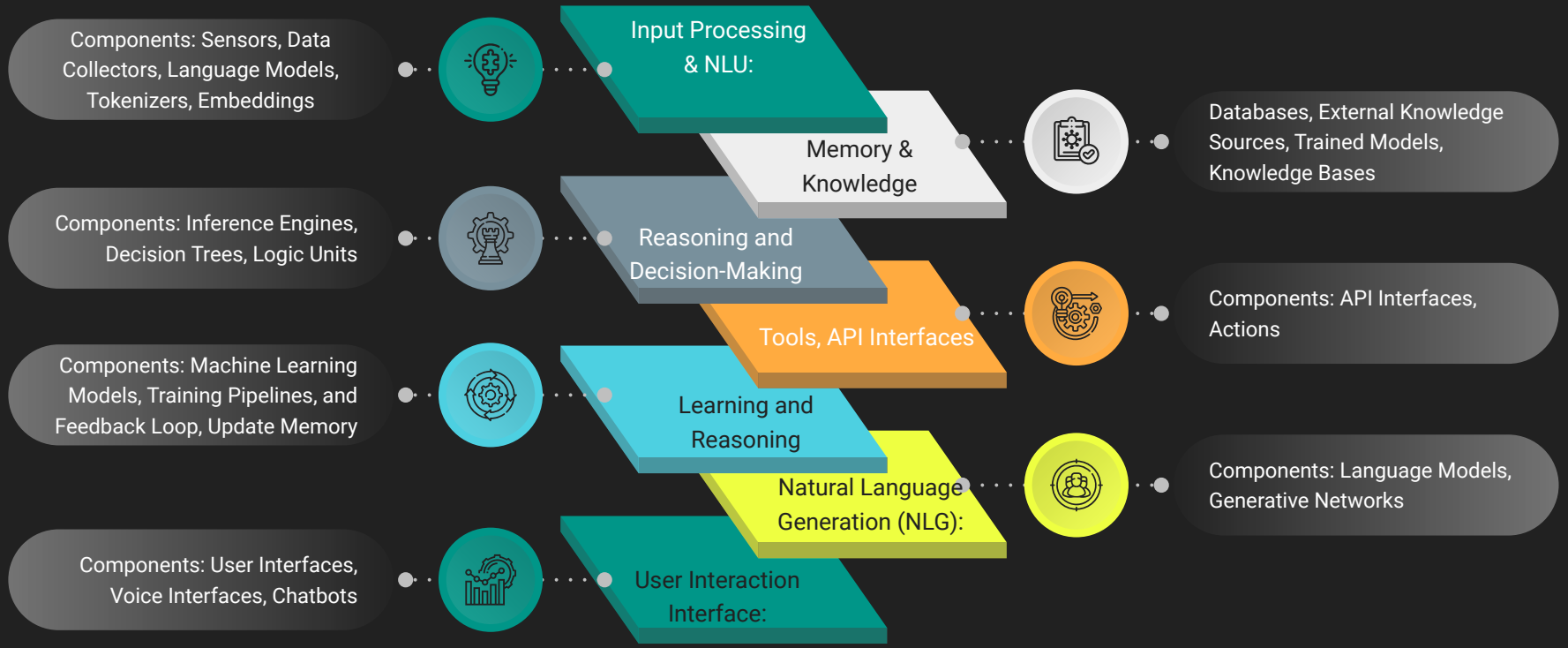
Contextual Awareness:They maintain context over the course of an interaction, allowing for more meaningful and relevant exchanges.

Interactive and Conversational Abilities: AI agents can engage in back-and-forth dialogue with users, providing interactive and dynamic responses.

Knowledge & Memory: They can integrate and apply a vast amount of knowledge from their training data to answer questions or solve problems.

AI Agents Components

Agent building blocks



Types of Agents (Traditional)

Various approaches to agent-based systems include:

Simple Reflex Agents: act solely based on the current percept ("sensor input" or "environmental data.") using condition-action rules (if-then rules) to respond to specific conditions in the environment. i.e A thermostat that turns on the heater if the temperature drops

Model-Based Reflex Agent: maintain an internal state that depends on the percept history, which helps them handle partially observable environments. Ex. A robot vacuum that maps the layout of a room to navigate more efficiently.

Goal-Based Agents : act to achieve specific goals. They consider future states and decide actions that can lead to goal achievement. Ex: A chess-playing agent that selects moves to checkmate the opponent.

Utility-Based Agents: These agents act to maximize their utility by considering the desirability of different states and make decisions that maximize their overall utility. Ex: A recommendation system that suggests products to maximize customer satisfaction and sales.

Learning Agents: These agents improve their performance over time by learning from experiences and consist of a learning element that modifies the agent’s behavior based on feedback and a performance element . Ex A speech recognition system that improves its accuracy as it processes more voice samples.

Types of Agents …continued

Knowledge-Based Agents:
Characteristics: Utilize a knowledge base and inference mechanisms to make decisions using logical reasoning. Ex: Expert systems, diagnostic systems in healthcare.

Reactive Agent:
Respond in real-time to environmental changes without extensive internal modeling or planning.Ex: Autonomous drones, simple robotic systems.

Proactive Agents:
Take initiative and perform actions to achieve goals, often anticipating future states and events. Ex: Personal assistant bots, intelligent scheduling systems.

Hybrid Agents:
Use different approaches, such as reactive and deliberative elements, to balance real-time responsiveness with strategic planning.: Advanced autonomous vehicles, ReAct agents.

Social Agents:
Interact with other agents or humans to achieve individual or collective goals, often using communication and collaboration. Ex:Multi-agent systems, collaborative robots (cobots).

Types of Agents …more recent

Embodied Agents:
Exist in a physical form and interact with the physical world, often requiring sensory and motor capabilities. Ex : Humanoid robots, autonomous drones.

Multi-Agent Systems (MAS):
Consist of multiple interacting agents, which can be either cooperative or competitive, to achieve complex tasks. Ex: Traffic management systems, distributed sensor networks.

Autonomous Agents:
Operate independently, making decisions and acting without human intervention.
Ex: Self-driving cars, autonomous industrial robots.

Adaptive Agents :
Adjust their behavior based on changes in the environment or their own experiences, often incorporating elements of learning. : Personalized learning systems, adaptive security systems.

ReAct Agents:
Integrate reasoning and acting often used for tasks requiring both strategic planning and immediate responses. Currently one of the most popular and exciting agents

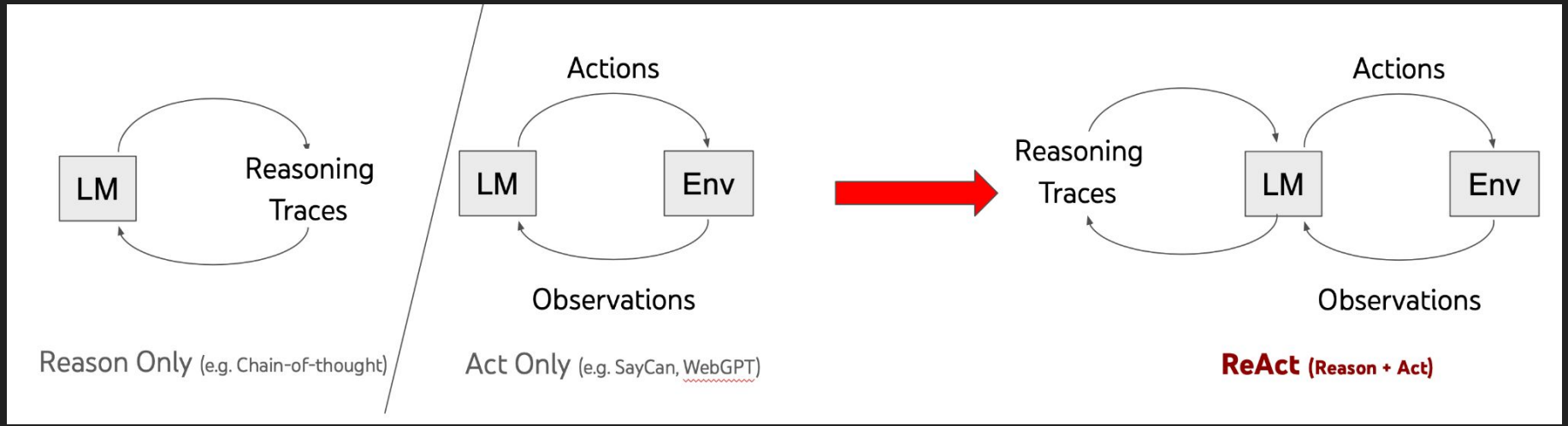
Features	LLM AI Agents	Traditional Machine Learning
Definition	Systems leveraging large language models (LLMs) for natural language tasks.	Algorithms and models designed to make predictions or decisions based on data.
Technical Expertise	No-code aside, requires expertise in programming (LLMs and data preparation. Python programing(No-code aside)	Requires expertise in machine learning algorithms, coding (No-code aside)
Data Dependency	Performance heavily depends on the quality and breadth of training data.	Relies on the quality of features and data; extensive preprocessing often required.
Development Speed	Rapid prototyping and deployment with minimal setup.	Requires significant time for data preprocessing, feature engineering, and model training.
Flexibility	Highly adaptable to new tasks with minimal retraining. Can adapt to some unseen situations	Often requires new models for different tasks. Can struggle with unseen data or distributions
Development Speed	Rapid prototyping and deployment with minimal setup.	Requires significant time for data preprocessing, feature engineering, and model training.
Transparency	Often seen as black boxes, difficult to interpret.	More interpretable with well-understood algorithms and model structures.
User-Friendliness	Interacts in natural language, reducing the need for specialized interfaces.	Requires specialized knowledge to develop and interpret models; user interfaces often need to be custom-built
Accuracy and Precision	Can vary; highly dependent on the quality of training data and model fine-tuning.	Typically high accuracy within the domain of the data and features used; often more precise for specific tasks.

ReaCt Agents

ReActAgent is designed to implement a particular type of AI agent that follows the ReAct (Reflect, Act) paradigm. This paradigm involves an agent that can reflect on its actions and the environment before deciding on the next action,

Reasoning: The agent analyzes the task at hand and formulates a plan. This involves considering past interactions, identifying missing information, or determining the best course of action.

Acting:The agent takes action based on its reasoning. This could involve: accessing external tools (like web search APIs) or Interacting with the environment or requesting clarification from the user



PAPER https://react-lm.github.io/

Chain of Thought and Chain of Reasoning

Chain of Thought (CoT) involves guiding an AI model to articulate its thought process in a step-by-step manner, similar to human problem-solving. This method helps the model to:

1. **Break Down Tasks**: Dived a complex problem into smaller, manageable steps, the model can handle intricate tasks more effectively.
2. **Transparency**: CoT makes the reasoning process of the model more transparent, allowing users to understand how the model arrives at a conclusion.
3. **Enhance Accuracy**: Encouraging the model to consider all necessary steps before providing an answer leads to more accurate results.

Chain of Reasoning (CoR) builds on CoT by emphasizing logical coherence and causal relationships between the steps. It ensures that:

1. **Logical**: Each step follows logically from the previous one, maintaining a clear **causal** relationship.
2. **Detailed Explanation**: CoR provides detailed explanations for each step, enhancing the interpretability and reliability of the AI's reasoning.
3. **Robustness in Complex Tasks**: CoR is particularly useful for tasks that require deep understanding and multiple layers of reasoning.

AI Agent Use Cases

Sentiment Analysis: Analyzing customer feedback to determine overall sentiment towards products or services.

Chatbots and Virtual Assistants: Powering conversational agents that can understand and respond to user queries in natural language.

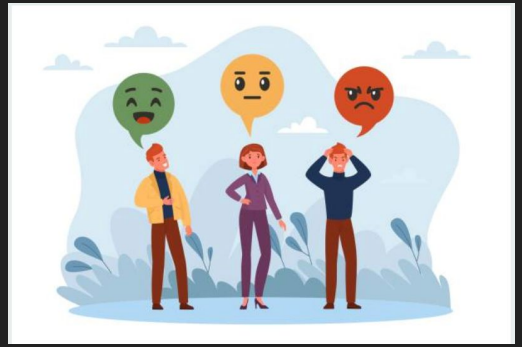
Machine Translation: Translating text or spoken language from one language to another, used in services like Google Translate.

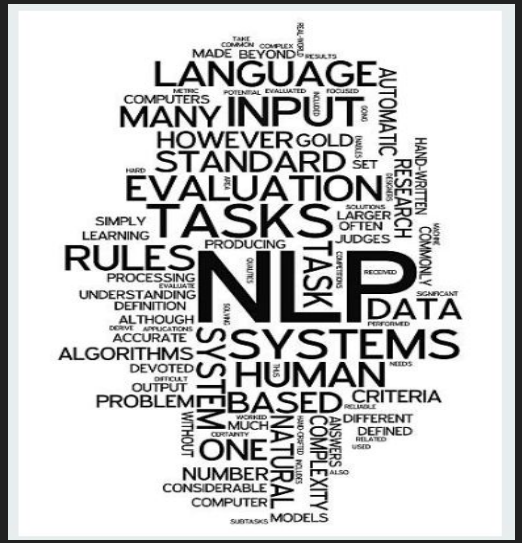
Speech Recognition: Converting spoken language into text, used in voice-activated assistants like Siri and Alexa.

Text Summarization: Automatically generating a concise summary of a long article or document.

Named Entity Recognition (NER): Identifying and classifying text into predefined categories, such as names of people, organizations, locations, etc.

Topic Modeling: Discovering the "topics" that occur in a collection of documents, helping in organizing, understanding, and summarizing large datasets of text.





Which LLM for You Agent

GPT-3.5 Turbo, GPT-4 Tubo, GPT-4o:

- **Developed by**: OpenAI. An advanced version of GPT-4 Turby, GPT-4o offers enhanced performance for general-purpose AI applications. GPT-3.5Turbo is OpenAI's fastest and most inexpensive model best for simple tasks

Llama 3:, 8B, 70B (Open Source)

- **Developer** by Meta AI is the latest in Meta's Llama series, Llama 3 comes in 8 billion and 70 billion parameter versions. Designed for scalability and real-time processing, making it suitable for applications requiring timely decision-making.

Gemini 1.5:and **Gemma** (Open Source)

- **Developed** by Google DeepMind. Gemini 1.5,features a significantly larger context window, allowing it to handle more extensive data inputs. **Gemma** is an open sourced, less capable model but received good reviews
- **Developer**: Google

Mistral 7B, 8X7B, 8x22B (Open Source)

- **Developed** Mistral AI. Mistral offers several models, including the Mixtral 8x22B, which uses a sparse mixture of experts to optimize performance and cost. It's suitable for high-demand applications such as real-time analytics and autonomous systems.

Claude 3, Opus, Sonnet, and Haiku:

- **Developed by**: Anthropic and Known for its focus on ethical AI, Claude 3 is used in applications like Slack, Notion, and Zoom. It excels in handling complex queries and providing safe, reliable interactions. Includes three models: Opus, Sonnet, and Haiku

Agent Tools

AI tools are specialized software components that empower AI and act as an AI agent's toolbox, The tools encompass a variety of functionalities that allow agents to collaborate and access information to complete complex tasks. some common types:

1. Information Retrieval Tools:

- These tools allow agents to access and process information from external sources. Examples include:
  - **Web Scraping Libraries:** Agents can extract data from websites for further processing.
  - **Search APIs:** Agents can query search engines like Google Custom Search or Bing Web Search to retrieve relevant information.
  - **Database, Vector Database, or Knowledge Graph:** These tools allow agents to access structured or unstructured information from knowledge repositories.

2. Communication and Interaction Tools:

- These tools facilitate communication between agents and with external systems. Examples include:
  - **API Integration Tools:** Agents can interact with various APIs like stocks APIs or map APIs to gather real-time data.
  - **Messaging Tools:** Agents can send messages to other agents within the chain or to external systems for further processing such as email, chat, message boards etc.

Agent Tools

3. Text Processing Tools:

- These tools allow agents to manipulate and analyze text data. Examples include:
  - **Tokenization:** Breaking down text into individual words or units.
  - **Named Entity Recognition:** Identifying and classifying entities within text (e.g., people, locations).
  - **Natural Language Processing (NLP) Libraries:** Providing advanced text analysis capabilities like sentiment analysis or summarization.

4. Content Generation Tools:

- These tools allow agents to generate different creative text formats. Examples include:
  - **Text Summarization:** Summarizing longer pieces of text for conciseness.
  - **Creative Text Generation:** Generating poems, scripts, or other creative text formats based on prompts.
  - **Machine Translation:** Translating text from one language to another.

5. Reasoning and Decision-Making Tools:

- These tools, still under development, aim to equip agents with reasoning capabilities. Examples (potential future tools):
  - **Causal Reasoning:** Understanding cause-and-effect relationships between events.
  - **Logical Reasoning:** Applying logical rules to draw conclusions.

Agent Chains for Complex Sequences

AI, agents are like intelligent assistants, constantly learning and interacting with their environment.

some tasks require more than just one agent. This is where AI agent CHAINS come in, combining the strengths of multiple agents to tackle complex challenges.

For Example: Research Agent

You work for a big 4 consulting firm. You need an AI assistant to summarize multiple research papers across various and generally an industry focused report. Depending on the topics surface, send a well crafted email to the relevant clients interested in the various industries well crafted email summarizing the attached report .

A single LLM agent might excel at summarizing text, but it might lack the knowledge to find relevant clients or properly structure a professional email and send it

Agent Chain Example: Research Agent

Information Gathering: The first agent in the chain, a "researcher" agent, might use web scraping or search APIs to find relevant academic papers.

Content Generation: This might be an LLM agent skilled in research summarization. It utilizes the information from the "researcher" agent to structure the reort and generate text sections.The chain culminates in a completed research paper, a collaborative effort by multiple specialized agents.

Email Report: The chain culminates in a completed summary research report, combined with a well crafted email to the client

initiates the process by submitting a request - “Summarize research papers

The "researcher" agent then shares the extracted information (topics, author background, citations, summaries) with the next agent in the chain.

Agents can interact and share information throughout the process. The "researcher" agent might identify gaps in the paper, prompting the LLM agent to generate additional content based on its findings.

Agent Chains for Complex Sequences

Benefits of AI Agent Chains:

Tackling Complex Tasks: By combining specialized agents, chains can handle tasks beyond the capabilities of a single agent.

Enhanced Efficiency: Different agents can work on their tasks simultaneously, potentially accelerating the completion process.

Improved Accuracy: Information sharing between agents allows for cross-checking and improved overall accuracy of the final output.

Modular Design: Chains can be easily adapted by adding or removing agents depending on the specific task and required skills.

Challenges and Considerations:

Communication Protocols: Clear communication protocols between agents are crucial for smooth information exchange and seamless collaboration.

Error Propagation: Errors from one agent can propagate through the chain, leading to issues in the final output. Robust error handling is essential..

AI Agent Chains : Frameworks

Some notable tools and frameworks used for building AI agent chains:

1. **LLama Index**
   - Provides tools and utility functions to build effective AI agent chains, including FunctionTool and QueryEngineTool for handling specific tasks. :Supports function calling and query engine integration and provides utility tools for handling large datasets and complex queries.
2. **LangChain**
   - A framework designed to build applications that involve complex interactions with large language models (LLMs).Supports chaining multiple LLM calls. Integrates external data sources and services.
3. **BabyAGI**
   - : An experimental AI agent framework that automates task completion through recursive self-improvement and planning.Features Autonomous task generation and prioritization. And Recursive task execution and improvement.
4. **Auto-GPT**
   - An autonomous AI agent framework that uses GPT-4 to generate and execute tasks without human intervention. Fetures goals and tasks generation based on a user's initial input. Executes tasks autonomously and generates new tasks based on progress.
5. **ReAct Agent**
   - A framework for building reactive AI agents that respond to real-time data and events. **Features**:Real-time decision making and can Integration with real-time data sources.

LangChain Framework

Framework designed to simplify the development of applications using large language models (LLMs). It provides tools and abstractions to create, manage, and deploy LLM-based applications efficiently.

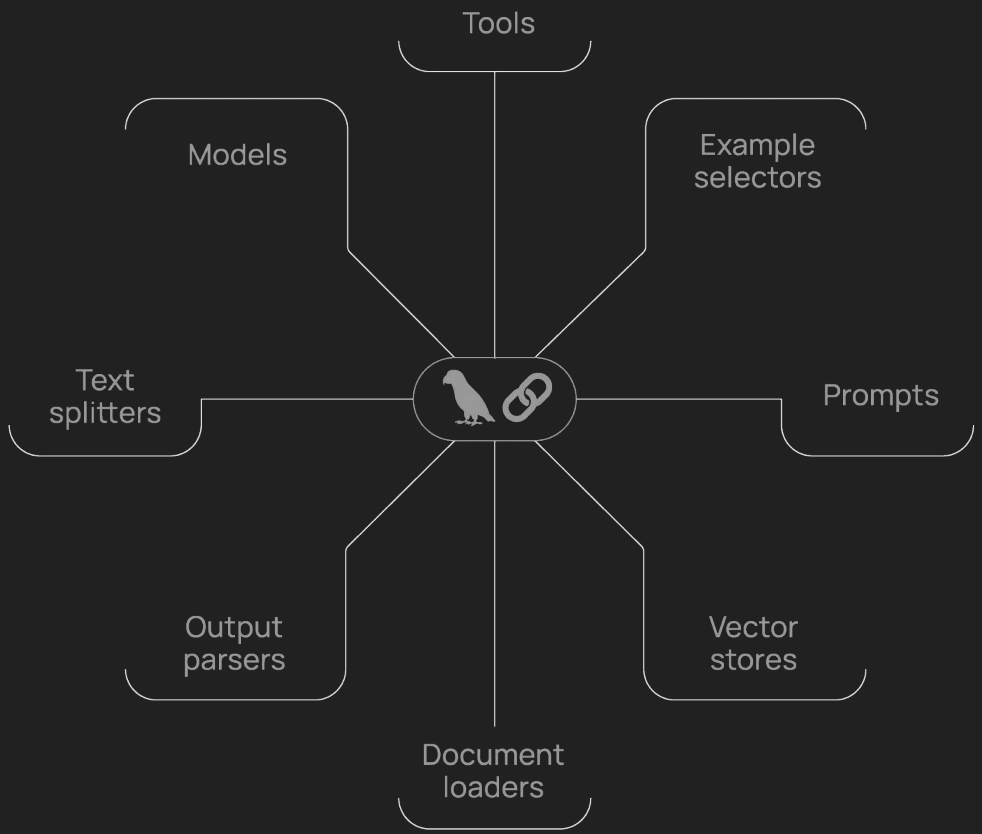
Key Features:

1. **Modular Design**: LangChain offers a modular approach, allowing developers to use one or more components
2. **Prompt Management**: Tools for creating, storing, and managing prompts
3. **Chains**: Facilitates the creation of sequences of actions or calls (chains) that an agent can follow to perform complex tasks.
4. **Memory**: Supports short-term and long-term memory for agents, enabling them to maintain context across interactions.
5. **Integration**: Seamless integration with various APIs and data sources, making it versatile for different applications.

Thus langchain Conversational Agents: Building chatbots and virtual assistants that can handle complex dialogues.

- **Automated Workflows**: Creating agents that automate multi-step workflows, such as data processing or customer support.
- **Content Generation**: Developing tools for generating articles, reports, or creative content based on user inputs.
- **Data Analysis**: Leveraging LLMs to interpret and analyze large datasets, providing insights and summaries.

LangChain Framework



LangChain streamlines the process of creating and managing AI agents by offering tools to handle LLM interactions effectively. Its modular approach allows developers to build sophisticated agents that can perform a variety of tasks, from simple responses to complex decision-making processes.

Contribution to AI Agents:

Customization: Highly customizable, allowing developers to tweak models, prompts, and chains to suit specific use cases.

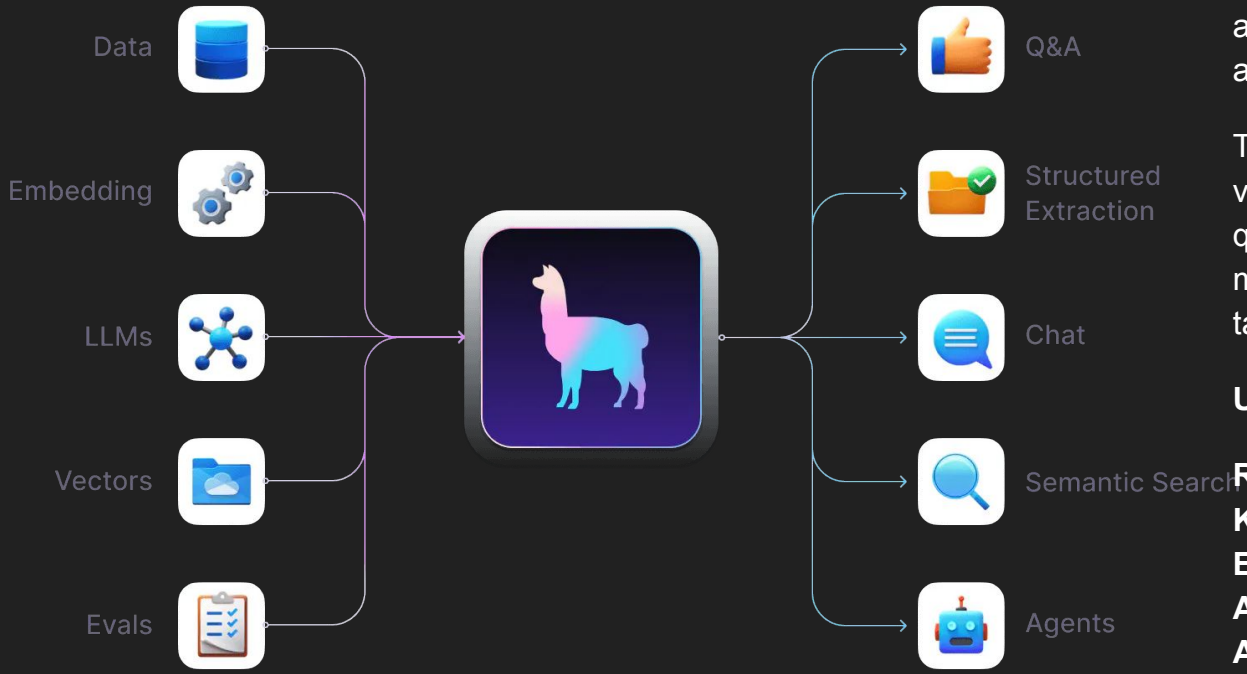
LamaIndex Famework

LlamaIndex (formerly known as GPT Index) is a framework designed to simplify the integration of large language models into applications, particularly focusing on data indexing and retrieval.

Key Features:

1. **Data Indexing**: Provides robust tools for indexing various data sources for efficient retrieval of large datasets.
2. **Query Optimization**: Enhances the process of querying indexed data using LLMs, making data access efficient & accurate.
3. **Integration with LMs**: Integration with popular LLMs, allowing for powerful data processing and interaction capabilities.
4. **Scalability**: Designed to handle large-scale data, making it suitable for enterprise-level applications.
5. **Custom Query Handlers**: for customization to meet specific application needs and improve response accuracy.

LlamaIndex Framework



LlamaIndex enhances the capability of AI agents by providing solid data indexing and retrieval mechanisms.

This allows agents to handle large volumes of information and respond fairly quickly to user queries, making them more effective in knowledge-intensive tasks.

Use Cases:

RAG system, Enterprise Search, Knowledge Management, **Search Engines**, **Customer Support**:, **Data Analysis**: and **Contribution to AI Agents:**

LangChain Vs LamaIndex

Both LangChain and LlamaIndex provide powerful tools for developing AI agents, each with a unique focus:

- **LangChain**: Emphasizes modular design, prompt management, and workflow automation, making it ideal for building conversational agents and complex automation tasks.

  Very popular but first few implementations were buggy. these bugs can cause unexpected behavior or errors when running agent chains. In turn agents chains can be difficult to debug. Other concerns include complex abstractions and obfuscated prompts

- **LlamaIndex**: Focuses on data indexing and retrieval, optimizing how agents interact with large datasets, and is particularly useful for knowledge management and search applications. Popular and robust

  Lacks the breadth and depth of langchain functionalities and tools. Specific details of how LlamaIndex indexes and retrieves data might not be entirely transparent (to me at the last use time)

Challenges of LLM AI Agents

While incredibly powerful, building and using LLM AI agents involve navigating a complex landscape of technical, ethical, and practical challenges. including:

1. **Integration Complexity**: Integrating multiple APIs, tools and services can be complex, requiring extensive customization **Latency**:High latency in response times can degrade the user experience, especially in real-time applications.
2. **Data Handling and Privacy**:Ensuring that sensitive user data is handled securely and in compliance with privacy regulations and integrating data from various sources while maintaining data integrity and consistency.
6. **User Experience**: Designing user-friendly interfaces that facilitate smooth interactions with the AI agent. New UX paradigm
7. **Maintenance and Updates**: Rregularly updating and maintaining the AI agents to fix bugs, improve performance, and adapt to new requirements, latest version. Ensuring that AI agents can continuously learn from new data and interactions
8. **Dependency on High-Quality Data**: High-quality, diverse datasets are essential for training effective AI agents. Poor data quality can lead to suboptimal performance.
9. **Responsible AI :** agents face hurdles in data quality, explainability, safety, and considerations like bias and fairness.
10. **Hallucinations and Inconsistent Outputs**: LLMs can sometimes generate outputs that are factually incorrect or nonsensical, often called hallucinations. This lack of consistency can make it difficult to trust the agent's responses.
11. **Limited Explainability**: Understanding how LLMs arrive at their outputs can be challenging. This "black box" nature makes it difficult to debug errors or identify the reasoning behind an agent's response.
12. **Limited User Contro**l: Users might have limited control over the information sources or reasoning processes used by agent.
13. **Identifying Trustworthy Responses:** It can be difficult for users to assess the accuracy and reliability of the information generated by the LLM agent.

RAG: Supercharging LLM Agents with Real-World Knowledge

RAG (Retrieval-Augmented Generation) Overview

Retrieval-Augmented Generation (RAG) is a framework that combines retrieval-based methods with generative models to create more accurate and contextually relevant responses. By integrating retrieval mechanisms, RAG enhances the generative capabilities of large language models (LLMs) like GPT-3.

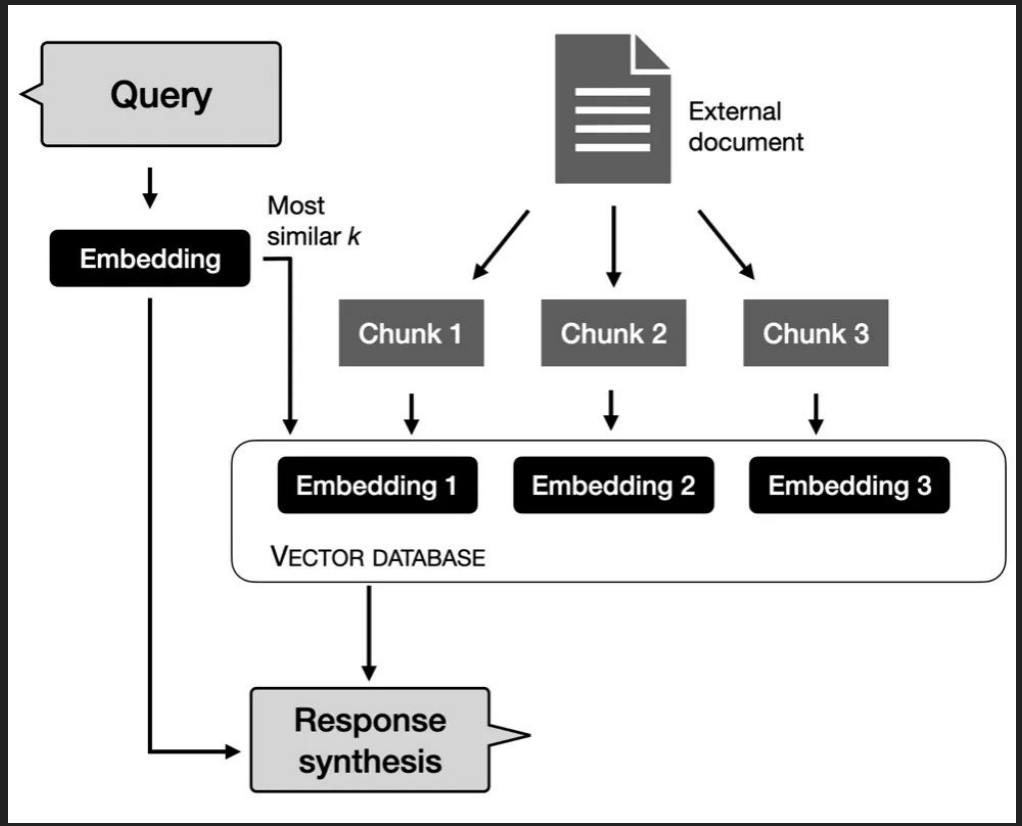
Key Features:

1. **Retrieval Component**: Retrieves relevant documents or data from a large corpus based on the input query.
2. **Generation Component**: Uses a generative model to produce responses, leveraging the retrieved documents to enhance accuracy and relevance.
3. **End-to-End Training**: The retrieval and generation components can be trained together, optimizing the entire pipeline for better performance.

Use Cases:

- **Question Answering**: Providing accurate answers by retrieving relevant documents and generating responses based on them.
- **Conversational Agents**: Enhancing chatbot responses with contextually relevant information retrieved from knowledge bases.
- **Content Creation**: Generating detailed and accurate content by incorporating relevant information from large datasets.

RAG: Vectors Embeddings



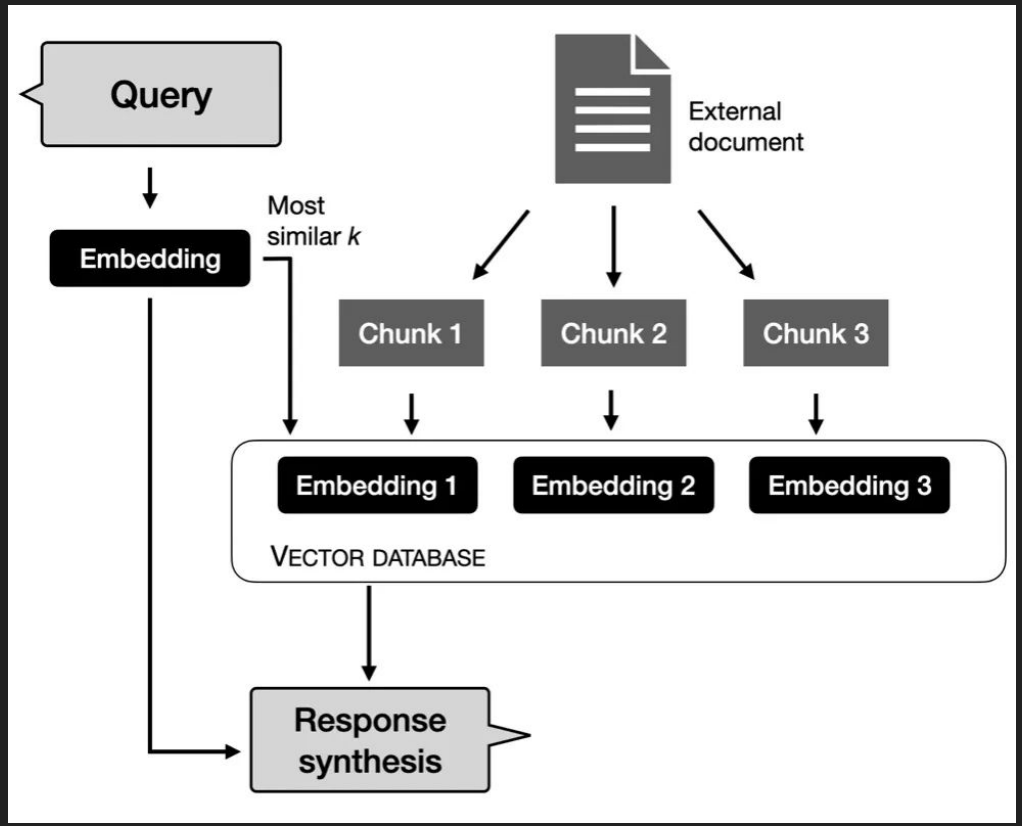
The **VECTORS** in RAG (and our llamaindex notebook example) represents a document or a chunk of text in a high-dimensional space.

Vector **EMBEDDINGS** are high-dimensional numerical representations of textual data. These embeddings capture the semantic meaning of the text, enabling the system to perform similarity searches effectively

So Imagine each word or concept within the text being assigned a numerical value along multiple dimensions.

This numerical representation captures the semantic meaning and relationships between words in the text.

RAG: Chunking



CHUNKING is the process of breaking down large documents or datasets into smaller, manageable pieces, or "chunks," which can then be indexed and retrieved more efficiently.

Key technique for enhancing the performance and accuracy of RAG.

Challenges:

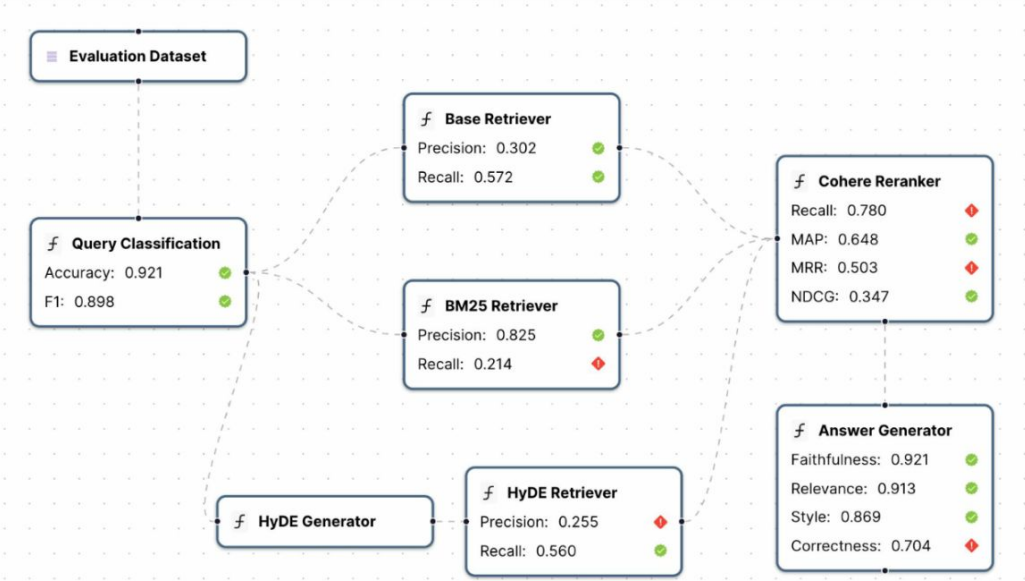
Chunks too small: May not capture enough context, leading to a loss of meaning and hindering the LLM's understanding of the retrieved information.
Chunks too large: Can be computationally expensive to process and might still contain irrelevant details, reducing retrieval precision.

RAG: Chunking

Chunk Size	Average Response Time (s)	Average Faithfulness	Average Relevancy
128	1.55	0.85	0.78
256	1.57	0.90	0.78
512	1.68	0.85	0.85
1024	1.68	0.93	0.90
2048	1.72	0.90	0.89

1. Modular Evaluation

- Production LLM app pipelines are **complex**, similar to AV stack
- Modular evaluation is **must-have to pinpoint issues**
- **Continuous-eval**, an open-source framework built for modularity



Source: github.com/relari-ai/continuous-eval

relari

Lesson 1: Hands-on Notebook

Lesson 2: Hands-on Notebook

Lesson 3: Hands-on Notebook

Addendum

Definition: Embedding

An embedding is a term used to describe a set of techniques aimed at capturing semantic (relationship, meaning, interpretation..) properties of words by representing them as low-dimensional, continuous vectors. These vectors are also known as "embeddings". Here's a deeper look at the concept:

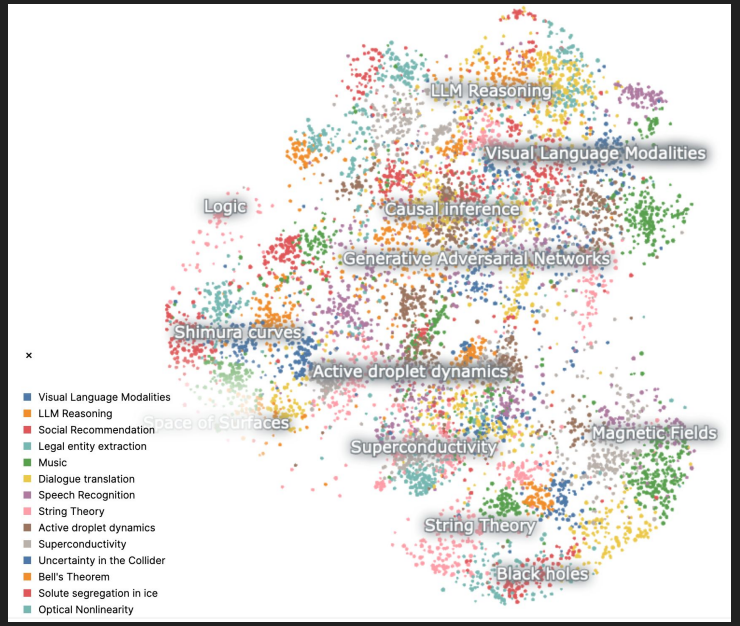
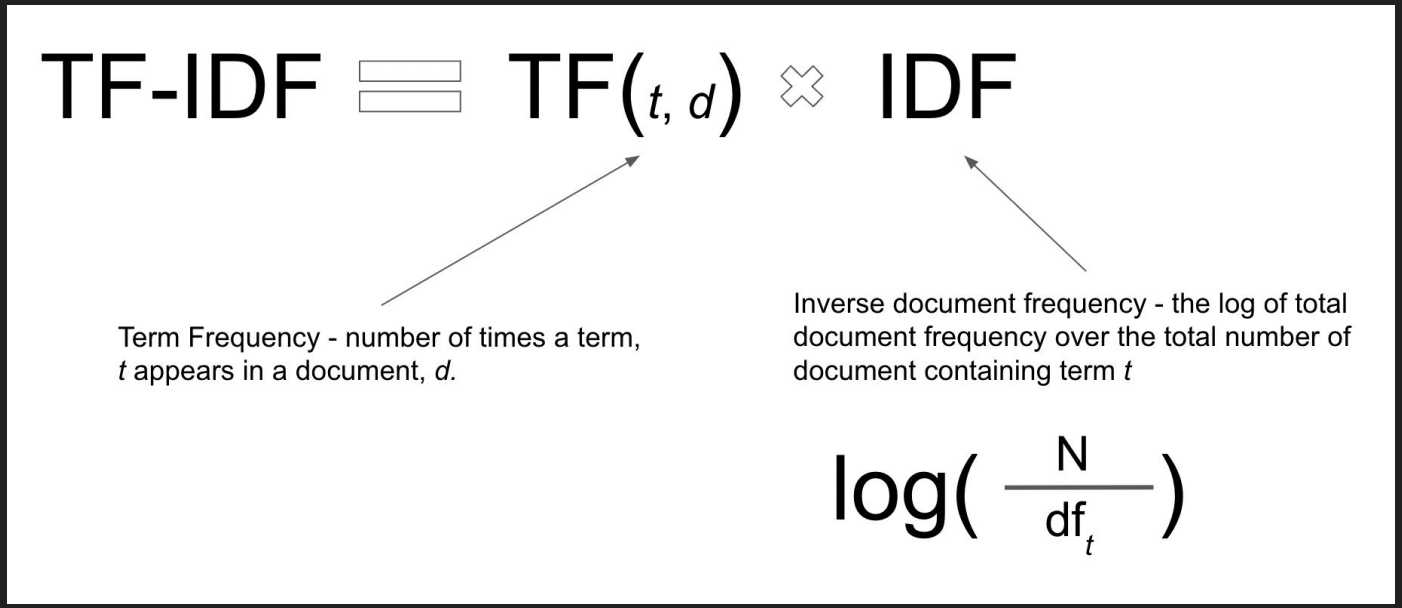


Image Credit: Nomic AI

Definition: Term Frequency & IDF

TF : How frequently a term appears in a document. The idea is to consider all terms equally important (however account for bias such as long documents). This is a Counting Based Embedding Techniques

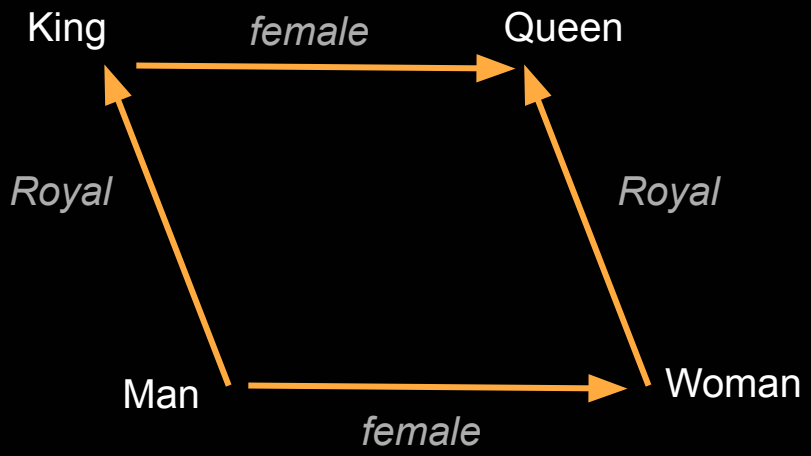
Inverse Document Frequency (IDF): This diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely.

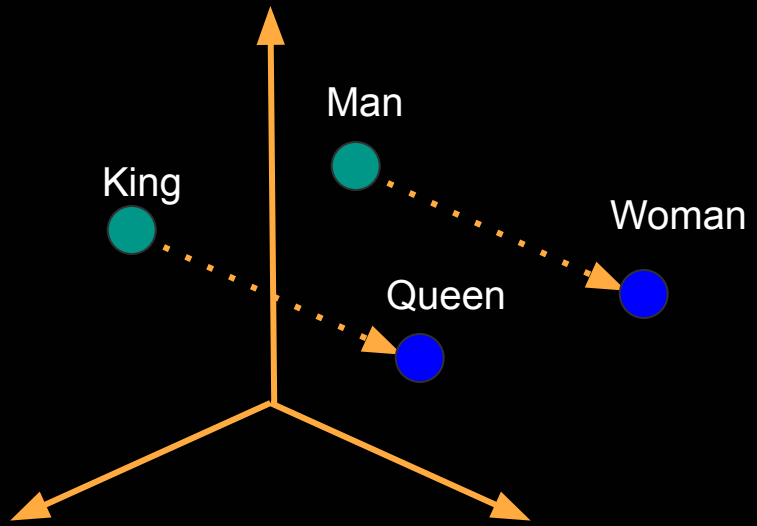


Word2Vec

Popular technique for generating dense word embeddings.

Word2Vec converts words from the vocabulary of a corpus into vectors of real numbers in a high-dimensional space. The core idea is to represent words in such a way that the spatial relationships between vectors capture semantic relationships between the words they represent, such as similarity and dissimilarity, contextual closeness, and so on.





Vectors and: One-Hot Encoding

Recall machine-learning algorithms operate on vectors of numbers

ONE-HOT ENCODING Convert categorical data variables into numerical format. This encoding helps algorithms better understand and process the dataset. This results in Sparse Vectors

- Create dummy variables.
- Each category is represented by a binary vector with all zeros
- except for a 1 at the index of the category (sparse Vectors)

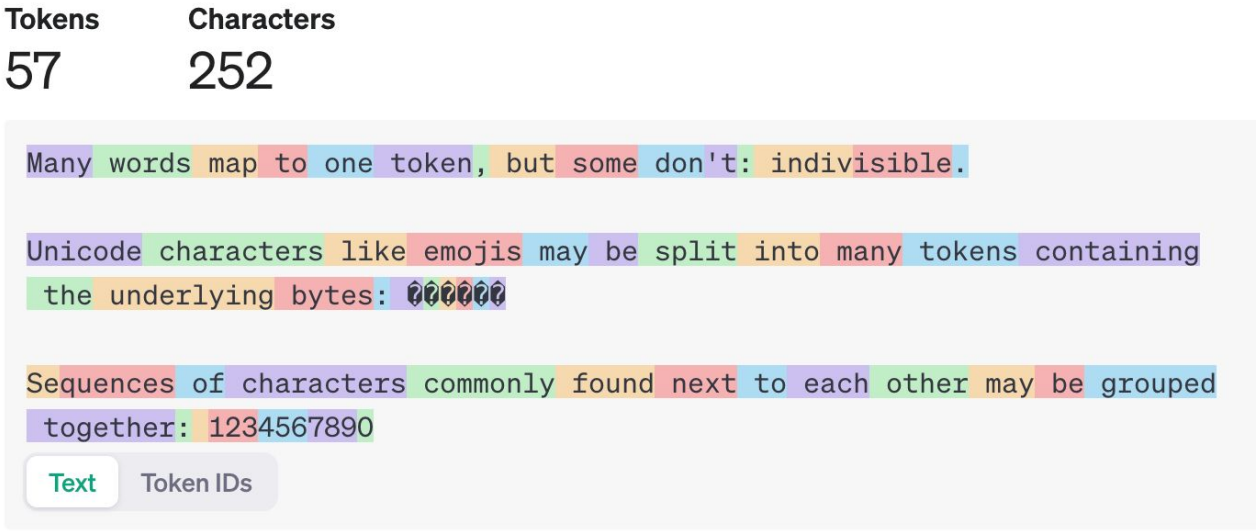
Feature (Color)
Red
Green
Yellow
Green
Red

One Hot Encoded Vector	Red	Green	Yellow
[1,00]	1	0	0
[0,1,0]	0	1	0
[0,0,1]	0	0	1
[0,1,0]	0	1	0
[1,00]	1	0	0

Sparse Vectors are not very effective (lots of zeros) so these are converted into Embeddings called Dense Vectors

Tokens

[Try OpenAI's Tokenizer]



A helpful rule of thumb is that one token generally corresponds to ~4 characters of text for common English text. This translates to roughly ¾ of a word (so 100 tokens ~= 75 words).

"This is a input text."

Tokenization

[CLS]	This	is	a	input	.	[SEP]
101	2023	2003	1037	7953	1012	102

Embeddings

0.0390, -0.0123, -0.0208, ...	-0.0558, 0.0151, 0.0031, ...	-0.0440, -0.0236, -0.0283, ...	0.0119, -0.0037, -0.0402, ...	0069, 0.0057, -0.0016, ...	0.0199, -0.0095, -0.0099, ...	-0.0788, 0.0202, -0.0352, ...

Embedding Layers

The one-hot vectors can be then fed into an embedding layer. This layer sparse vectors into high-dimensional (dense) vectors. These vectors capture semantic information, meaning words with similar contexts are represented by vectors that are close to each other in the vector space. Using embeddings we can process language based on the similarity of word meanings and their context.

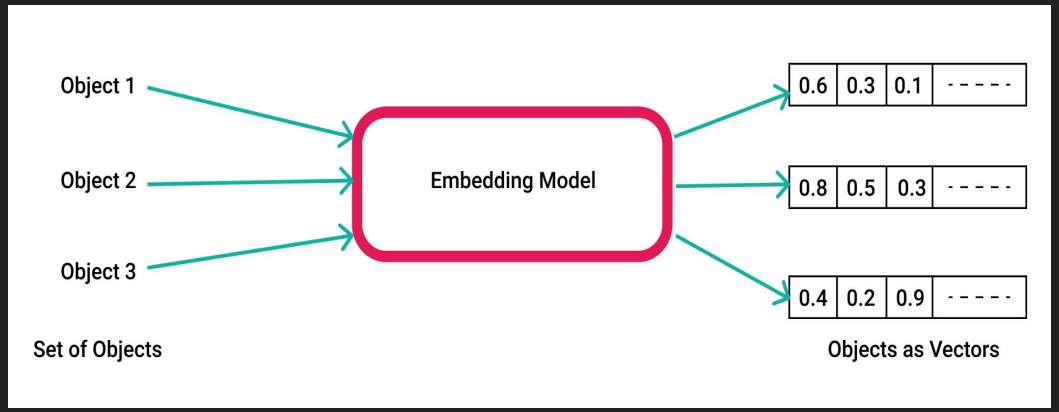
Tokenization: The process begins by breaking down input text into manageable pieces, known as tokens. Tokens can be words, parts of words, or even characters etc

Vectorization: Each token is then mapped to a vector in a high-dimensional space (each word). These vectors are learned during the training phase of the model, allowing the model to capture the nuances of language.

Semantic Relationships: Through training on huge datasets, embedding vectors learn to encode semantic relationships between words and contextual relevance.

Embedding Layers in NLP

Vector embeddings are just lists of numbers to allow machine learning algorithms to perform various operations with them. A whole paragraph of text or any other object can be reduced to a vector.



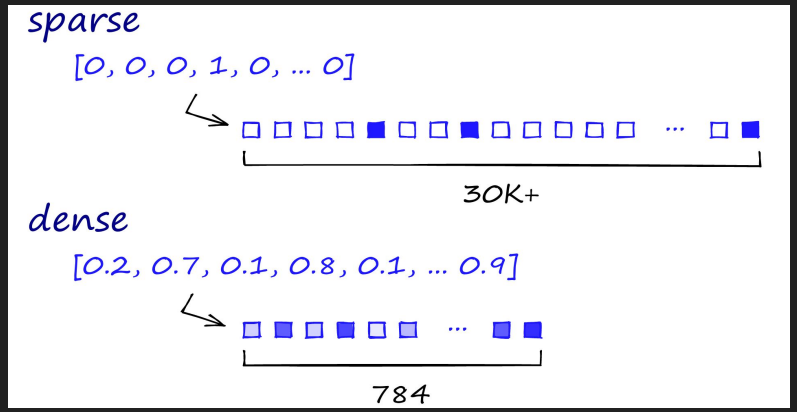
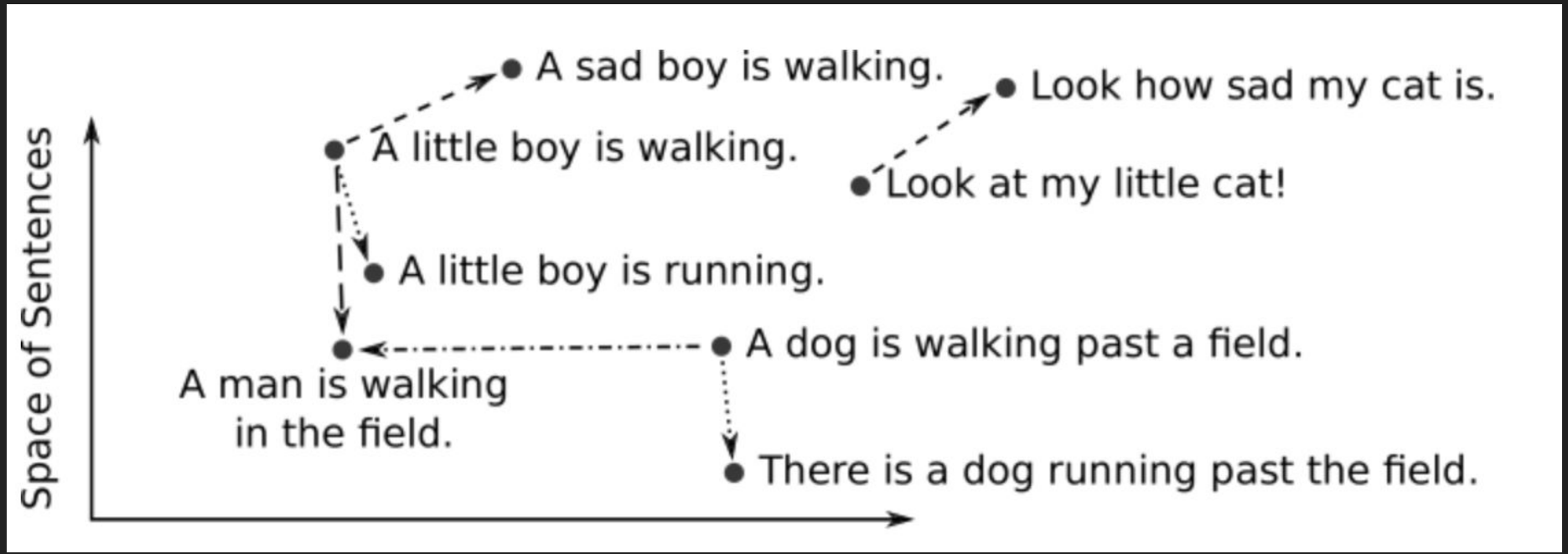


Image credit: https://www.pinecone.io/learn/series/nlp/dense-vector-embeddings-nlp/

Semantic Similarity

Vectors representation makes it possible to translate semantic similarity as perceived by us to proximity in a vector space. The semantic similarity of these objects and concepts can be quantified by how close they are to each other as points in vector spaces.



Source: https://www.pinecone.io/learn/vector-embeddings/

Embedding Layers Example

Tokenization: The model breaks down the input into tokens and then vectors: ["3 bedrooms", "2 baths", "ocean view", "newly renovated"].

"3 bedrooms": [1, 0, 0, 0] "2 baths": [0, 1, 0, 0] "ocean view": [0, 0, 1, 0 "newly renovated": [0, 0, 0, 1]

Vectorization: Each token is converted into an embedding vector. For example, "ocean view" might be closely related to other terms like "beachfront" in the model's learned vector space.

"3 bedrooms" → [0.85, -0.24, 0.15, 0.67, 0.32] (5 dimensions)

Semantic Relationships:

As the model processes these vectors through its layers, it uses the semantic information encoded in the embeddings to generate a contextually rich property description. For example, knowing the appeal of "ocean view" properties, the model might generate: "Enjoy breathtaking ocean views from this newly renovated home, featuring 3 spacious bedrooms and 2 modern bathrooms."

Step 2: Positional Encoding

Traditional neural networks struggle to understand the order of elements in a sequence and and transformer models don't inherently process sequential data in order. Positional encodings are added to the input embeddings to provide information about the position of each token in the sequence.

Ocean view: [0.9, -0.1, 0.3]

Positional encodings address this by injecting information about the relative position of each element (word) in the sequence. This allows the LLM to grasp the context and meaning even without relying solely on word order. Think of it like adding numbers to each word to indicate its place in the line.

Position Encoding(for "Ocean view"): [0.2, 0.2, 0.2]

This understanding is crucial in contexts where the arrangement of words alters the meaning or emphasis of a message, such as in real estate listings, where the placement of features can significantly impact a property's appeal.

Positional Encoding Example

Generate a property description emphasizing proximity to amenities. The sentence "Located in a vibrant neighborhood, the apartment is just a short walk from popular restaurants and convenient subway stops."

The order in which features are mentioned can influence a potential buyer's interest.

- “vibrant neighborhood" sets a lively context,
- followed by the convenience of "popular restaurants"
- "Close to subway stops" enhances the property's appeal.

Embedding without positional ending

vibrant neighborhood": [0.9, -0.1, 0.3] popular restaurants": [0.8, 0.2, -0.5]

The model understands that the mention of "a vibrant neighborhood" at the beginning is strategically placed to catch attention, and "a short walk from popular restaurants and convenient subway stops" highlights essential amenities

Psitional encoding: Position 2 (for "popular restaurants"): [0.1, 0.1, 0.1]

Embedding with Psitional encoding: popular restaurants": [0.9, 0.3, -0.4]

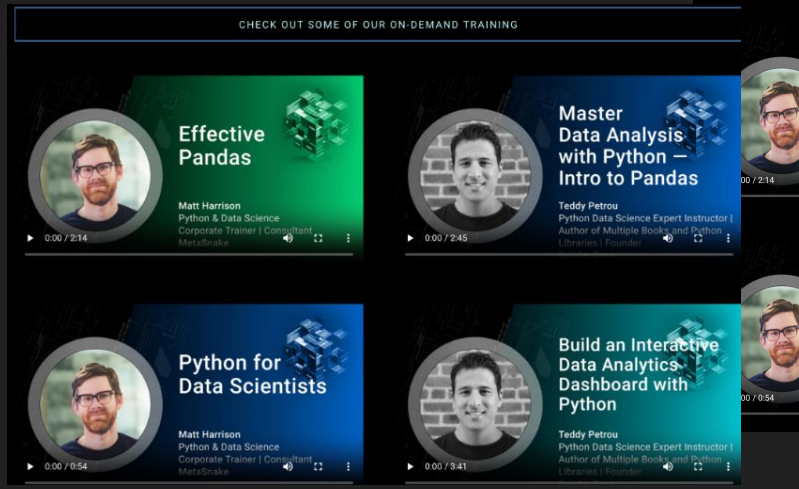
AI+ Additional Sources

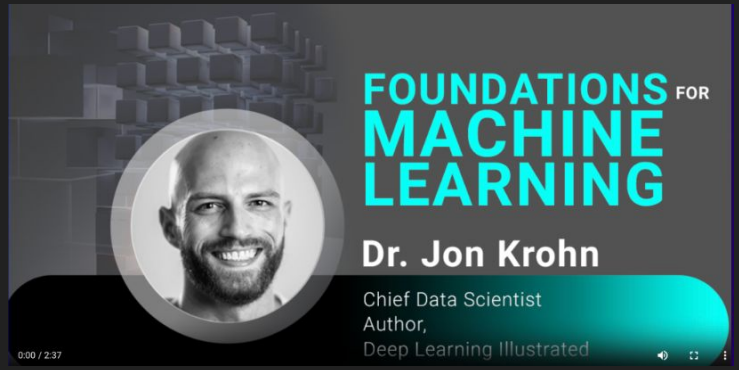
[East 2024 Bootcamp Bundle]

[Foundations in math, computer science and probability]

Deep Learning with Jon Krohn

[Generative Ai and Large Language Models]





THANK YOU