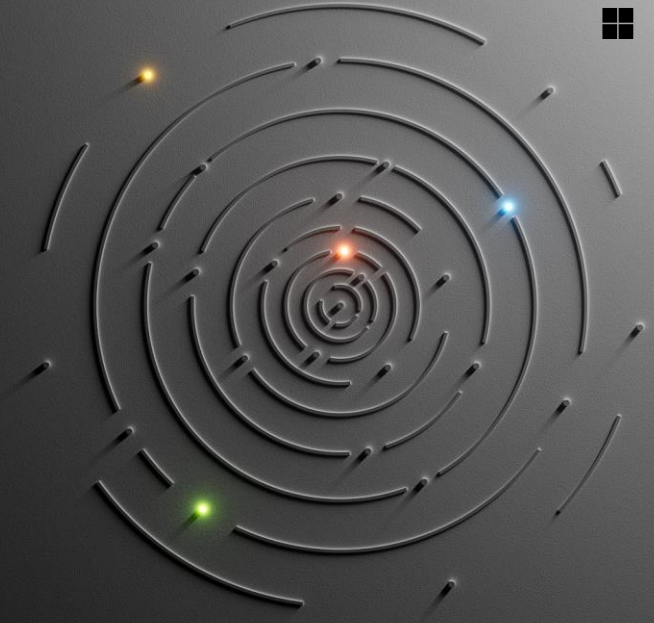




Generative AI

Concepts, Tools, and Applications
for Data Professionals

Buck Woody
Applied Data Scientist, Microsoft



Generative AI is a branch of artificial intelligence that can create new data or content by learning from existing content. It has applications in various domains such as text, image, audio, video, and code generation. In this session, you will learn how generative AI works, its limitations, the main tools and processes you can use, and where you can use it to enhance your data estate and data projects. You will also explore examples of how generative AI can help you with code generation, debugging, and optimization tasks. This session is suitable for data professionals who have some familiarity with machine learning and want to learn more about the potential and effective uses of generative AI.

This session is One part college lecture, one part history lesson, and one part career technology focus



Agenda

1. Introduction to AI and ML
2. Overview of NLP and GPT
3. Data Integration for AI and GPT
4. Q & A

- Introduction
 - What is generative AI and why is it important for data professionals?
 - What are the main types of generative AI models and techniques?
 - What are the main challenges and limitations of generative AI?
- Generative AI for Text
 - How to use natural language processing and natural language generation to create and manipulate text data
 - Examples of text generation tasks such as summarization, paraphrasing, translation, and content creation
 - Demo: Using a pre-trained generative model to generate text
- Generative AI for Code
 - How to use code analysis and code synthesis to create and improve code data
 - Examples of code generation tasks such as code completion, code documentation, code debugging, and code optimization
 - Demo: Using a pre-trained generative model to generate code
- Conclusion
 - Summary of the main points and takeaways from the session
 - Q&A and feedback from the audience



Framing

Our goals today:

- Understand Generative AI in the Context of AI History
- Gain an overview of GPT processes and designs
- Learn how to use GPT in a Data Context

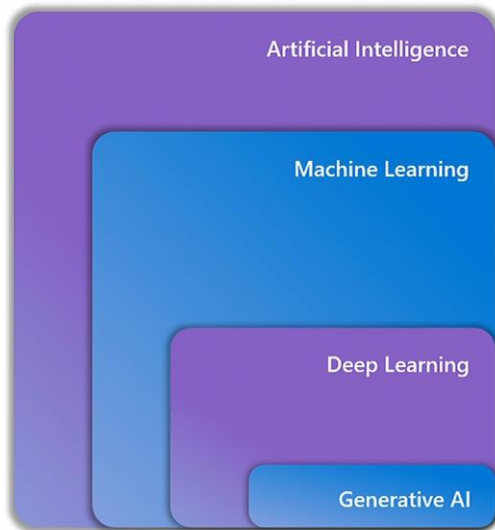


Introduction to AI

The progression of Artificial Intelligence

* Some great cheat-sheets for the concepts here - <https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-science-pdf-f22dc900d2d7>

Good overview - <https://www.geeksforgeeks.org/artificial-intelligence-an-introduction/#>



1956

Artificial Intelligence

the field of computer science that seeks to create intelligent machines that can replicate or exceed human intelligence

This image is made available under the Creative Commons CC0.1.0 Universal Public Domain Dedication

AI: Distinguished by input systems, any mathematical technique that mimics human behavior

ML: Using combinations of mathematical formulae to act over data, improving a prediction or classification from that data

DL: Using specialized layers of algorithms to train a model through the layers to create a prediction or classification



Artificial Intelligence

“Human intelligence can be so precisely described that a machine can be made to simulate it.”

Dartmouth Summer Research Project on Artificial Intelligence, 1956

Development

Symbolic

Logic
Fuzzy Systems Simulation
Expert Systems
Evolutionary Computing

Statistical

Inference Information Theory
Hidden Markov Models
Normative Bayesian Decision Theory

- <https://time.com/6271657/a-to-z-of-artificial-intelligence/>

An “Expert System” is where you ask an expert how they do their job, and then you codify that. It’s a kind of a series of “If-Then-Else” trees.

Then AI became more algorithmic – beginning to model things in math.

Formal definition and history: <https://web.archive.org/web/20071012025921/http://www.acm.org/class/1998/l.2.html>



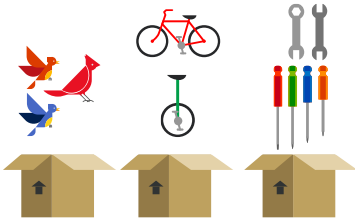
Natural Language Processing and Generative Pre-trained Transformers

Building on Predictions, Algebra, Calculus, Statistics for Deep Learning

Discriminative versus Generative - <https://www.assemblyai.com/blog/introduction-generative-ai/>

Machine Learning Uses and Algorithm Families

Which category
(Classification)



How much/many
(Regression)



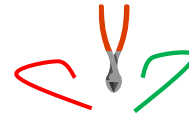
Which group
(Clustering, Recommender)



Is it odd
(Anomaly)



Which action
(Reinforcement Learning)



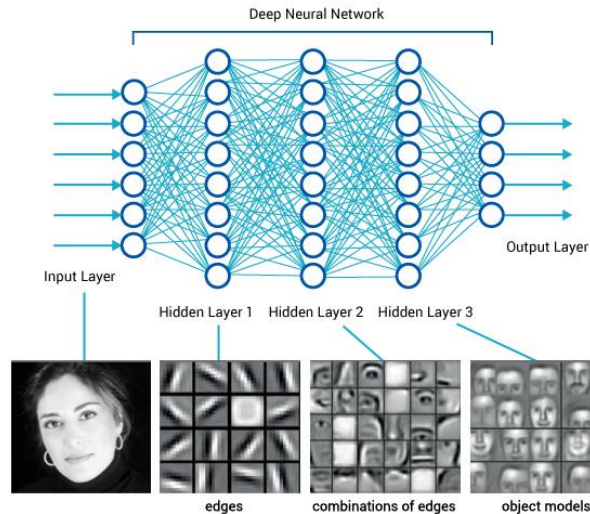
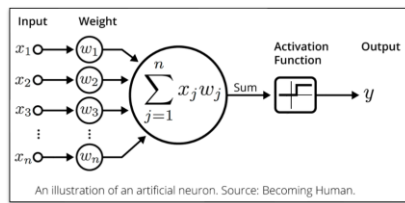
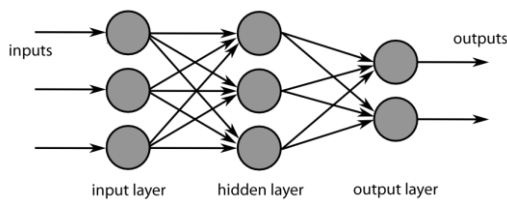
* Algorithm Types - <https://www.coursera.org/articles/machine-learning-algorithms>

Machine learning is a way of making computers learn from data without explicitly telling them what to do. It is a branch of artificial intelligence that uses algorithms and statistical models to find patterns and make predictions. Some examples of machine learning applications are web search, self-driving cars, speech recognition, and recommendation systems. There are some key concepts that you need to know to understand machine learning better. Here are some of them:

- **Data:** Data is the raw information that machine learning algorithms use to learn from. Data can be in different forms, such as numbers, text, images, audio, or video. Data can also be structured or unstructured, depending on how it is organized and labeled.
- **Algorithm:** An algorithm is a set of rules or instructions that a computer follows to perform a task. In machine learning, an algorithm is a process that learns from data and produces a model.
- **Model:** A model is the result of a machine learning algorithm. It is a representation of what the algorithm has learned from the data. A model can be used to make predictions or classifications on new data.
- **Training:** Training is the process of feeding data to a machine learning algorithm and adjusting the parameters of the model to improve its performance. Training can be supervised, unsupervised, or semi-supervised, depending on whether the data has labels or not.
- **Testing:** Testing is the process of evaluating the performance of a machine learning model on new data that it has not seen before. Testing can be done by using metrics such as accuracy, precision, recall, or F1-score, depending on the type of problem.
- **Feature:** A feature is a characteristic or attribute of the data that is relevant for the machine learning task. For example, if you want to classify images of animals, some features could be the color, shape, size, or texture of the animal. Features can be extracted from raw data or engineered by using domain knowledge.
- **Hyperparameter:** A hyperparameter is a parameter that controls the behavior of a machine learning algorithm. For example, the number of layers in a neural network, the learning rate, or the regularization factor are hyperparameters. Hyperparameters are not learned from data but are set by the user before training.
- **Overfitting:** Overfitting is a problem that occurs when a machine learning model learns too much from the training data and fails to generalize well to new data. Overfitting can be caused by having too many features, too complex models, or too little data. Overfitting can be prevented by using techniques such as cross-validation, regularization, or dropout.
- **Underfitting:** Underfitting is a problem that occurs when a machine learning model learns too little from the training data and fails to capture the underlying patterns or relationships in the data. Underfitting can be caused by having too few features, too simple models, or too much noise in the data. Underfitting can be improved by using techniques such as feature selection, feature engineering, or more complex models.

Deep Learning

Learning through layers



Images made available under the Creative Commons CC0 1.0 Universal Public Domain Dedication

Deep learning is a subfield of machine learning that uses artificial neural networks to learn from data. Neural networks are composed of layers of interconnected nodes that can perform complex computations and transformations on the input data. Deep learning is called "deep" because it uses multiple layers of neural networks, each with its own parameters and functions, to create a hierarchical representation of the data. Some key concepts that you need to know to understand deep learning better are:

- **Input layer:** The input layer is the first layer of a neural network that receives the raw data. The input layer can have different shapes and sizes depending on the type and dimensionality of the data. For example, if the data is an image, the input layer can be a matrix of pixels with three channels (red, green, and blue).
- **Hidden layer:** A hidden layer is any layer of a neural network that is not the input or output layer. A hidden layer can have different types of nodes, such as fully connected, convolutional, recurrent, or attention. A hidden layer can also have different activation functions, such as sigmoid, tanh, relu, or softmax. A hidden layer can perform various operations on the input data, such as filtering, pooling, encoding, decoding, or generating.
- **Output layer:** The output layer is the last layer of a neural network that produces the final result. The output layer can have different shapes and sizes depending on the type and dimensionality of the output. For example, if the output is a classification, the output layer can be a vector of probabilities with one node for each class.
- **Backpropagation:** Backpropagation is a technique that allows a neural network to learn from its errors and update its parameters accordingly. Backpropagation works by calculating the gradient of the loss function with respect to each parameter of the network using the chain rule. The gradient indicates how much each parameter contributes to the error and in which direction it should be adjusted. Backpropagation then applies an optimization algorithm, such as gradient descent or Adam, to update the parameters using the gradient and a learning rate.
- **Deep learning framework:** A deep learning framework is a software library that provides tools and functions for building, training, and testing neural networks. Some examples of popular deep learning frameworks are TensorFlow, PyTorch, Keras, and MXNet. A deep learning framework can simplify and speed up the development process by providing predefined layers, models, datasets, and metrics.

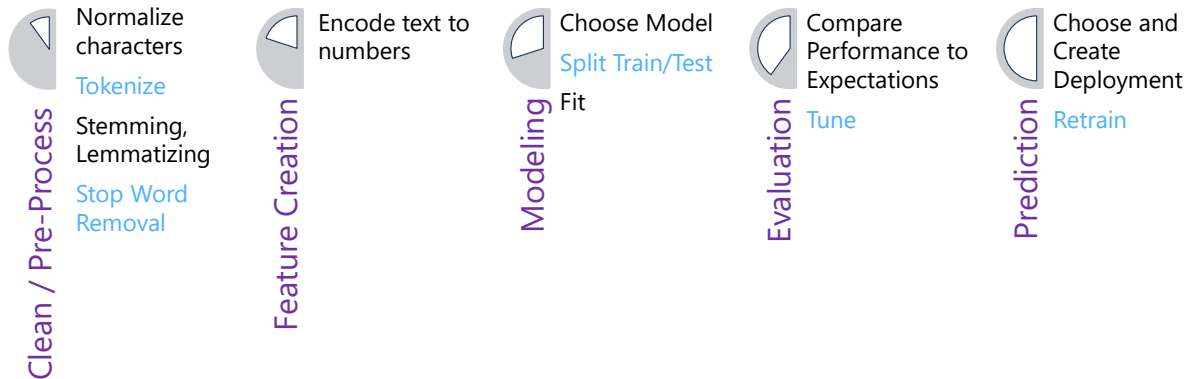
As an example, a machine learning workflow starts with relevant features being manually extracted from images. The features are then used to create a model that categorizes the objects in the image. With a deep learning workflow, relevant features are automatically extracted from images. In addition, deep learning performs "end-to-end learning" – where a network is given raw data and a task to perform, such as classification, and it learns how to do this automatically.

Another key difference is deep learning algorithms scale with data, whereas shallow learning converges. Shallow learning refers to machine learning methods that plateau at a certain level of performance when you add more examples and training data to the network.

A key advantage of deep learning networks is that they often continue to improve as the size of your data increases.



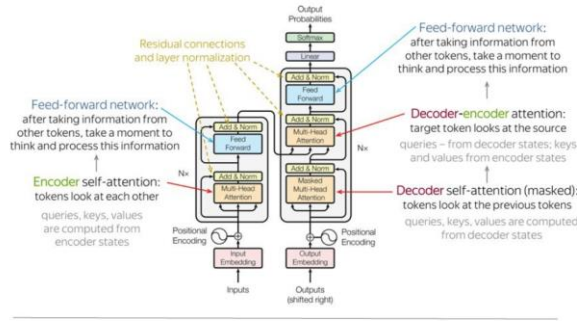
Natural Language Processing



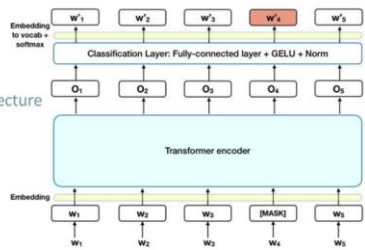
Natural language processing (NLP) is a branch of artificial intelligence that deals with the interaction between computers and human languages. NLP aims to enable computers to understand, analyze, generate, and manipulate natural language texts or speech. Some examples of NLP applications are chatbots, machine translation, sentiment analysis, and text summarization. There are some key concepts that you need to know to understand NLP better. Here are some of them:

- **Tokenization:** Tokenization is the process of breaking down a text or speech into smaller units called tokens. Tokens can be words, sentences, paragraphs, or symbols. Tokenization is the first step in many NLP tasks, as it helps to simplify and standardize the input data.
- **Parsing:** Parsing is the process of analyzing the structure and meaning of a text or speech. Parsing can involve different levels of analysis, such as syntactic, semantic, or pragmatic. Parsing can help to identify the parts of speech, the grammatical relations, the named entities, the sentiment, or the intention of a text or speech.
- **Embedding:** Embedding is the process of representing a text or speech as a vector of numbers in a high-dimensional space. Embedding can capture the semantic and syntactic features of a text or speech, as well as its similarity or dissimilarity with other texts or speeches. Embedding can be done by using different methods, such as word2vec, GloVe, BERT, or GPT.
- **Generation:** Generation is the process of producing a text or speech from a given input or context. Generation can involve different techniques, such as rule-based, template-based, or neural network-based. Generation can be used for various purposes, such as summarizing, translating, answering, or creating content.
- **Evaluation:** Evaluation is the process of measuring the quality and performance of an NLP system or model. Evaluation can be done by using different metrics, such as accuracy, precision, recall, F1-score, BLEU, ROUGE, or METEOR, depending on the type and goal of the task.

Transformer Architecture



BERT Architecture



GPT Architecture

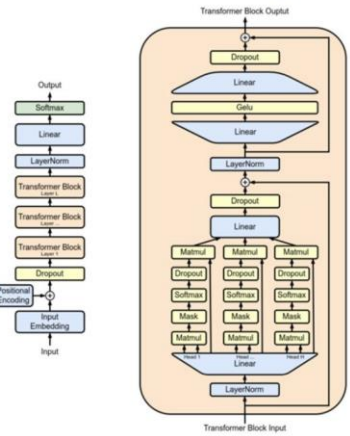


Image made available under the Creative Commons CC0 1.0 Universal Public Domain Dedication

Ashish Patel - Principal Research Scientist • ashishpatel.ca.2011@gmail.com

Ahona Saha - Machine Learning Scientist • ahonasa@gmail.com (updated: 2023-02)

Large language models are machine learning models that are trained on a large amount of text data to learn the patterns and structures of natural language. Large language models can perform various natural language processing tasks, such as text generation, text summarization, question answering, and sentiment analysis, by using a single model and without requiring any task-specific training. Some key concepts that you need to know to understand large language models better are:

- **Pre-training:** Pre-training is the process of training a large language model on a general-purpose text corpus, such as Wikipedia, books, news articles, or web pages. Pre-training aims to capture the common knowledge and linguistic features of natural language and store them in the model's parameters.
- **Fine-tuning:** Fine-tuning is the process of adapting a pre-trained large language model to a specific task or domain by using a smaller amount of task-specific or domain-specific data. Fine-tuning aims to adjust the model's parameters to optimize its performance on the target task or domain.
- **Self-attention:** Self-attention is a mechanism that allows a large language model to learn the relationships and dependencies between different parts of the input text. Self-attention works by computing a score for each pair of tokens in the input text based on their similarity and relevance. Self-attention then uses these scores to create a weighted representation of the input text that captures its context and meaning.
- **Transformer:** Transformer is a type of neural network architecture that is based on self-attention and consists of two main components: an encoder and a decoder. The encoder takes the input text and encodes it into a sequence of vectors using self-attention and feed-forward layers. The decoder takes the encoded vectors and generates the output text using self-attention, feed-forward, and output layers. Transformer is the backbone of many large language models, such as BERT, GPT, and T5.
- **Scaling:** Scaling is the process of increasing the size and complexity of a large language model by using more data, more parameters, more layers, or more computational resources. Scaling aims to improve the performance and capabilities of a large language model by enabling it to learn more diverse and nuanced aspects of natural language.

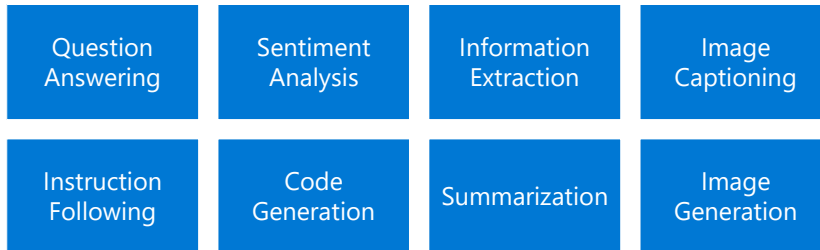
References:

- Very thorough resource - <https://writings.stephenwolfram.com/2023/02/what-is-chatgpt-doing-and-why-does-it-work/>
- Main Concepts - <https://willthompson.name/what-we-know-about-llms-primer>
- Main Concepts - <https://arstechnica.com/science/2023/07/a-jargon-free-explanation-of-how-ai-large-language-models-work/>
- Embeddings - <https://txt.cohere.com/embedding-archives-wikipedia/>
- Transformers - <https://jalammar.github.io/illustrated-transformer/>
- Transformers - <https://jalammar.github.io/illustrated-gpt2/>
- Attention - <https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf>
- Feedback - <https://arxiv.org/pdf/2203.02155.pdf>
- Mixture of Experts - <https://machinelearningmastery.com/mixture-of-experts/>

What is a foundation model?

This term was coined by Stanford University: <https://crfm.stanford.edu>

Train one model on a huge amount of data and adapt it to many applications



- <https://blogs.nvidia.com/blog/2023/03/13/what-are-foundation-models/>

Stanford University invented the term “foundation models” in their paper which was published on August 16, 2021: “On the Opportunities and Risks of Foundation Models”.

Paper: <https://arxiv.org/abs/2108.07258v3> (CRFM in the link is “Center for Research on Foundation Models”.)

Foundation models

- Extraction
- Summarization
- Generation (Text/Image/Sound)

More information at https://en.wikipedia.org/wiki/Foundation_models

And <https://renaissancerachel.com/understanding-foundation-models/>

What are foundation models?

✓ Searching for: **foundation models**

✓ Generating answers for you...

Foundation models are a paradigm for building AI systems that can be adapted to many applications¹. They are trained on a large amount of unlabeled data at scale, usually by self-supervised learning, and learn useful patterns that can be transferred to downstream tasks with better sample efficiency². Foundation models have shown impressive capabilities in language, vision, robotics, reasoning, and human interaction, and have been applied to various domains such as law, healthcare, and education¹. However, they also pose significant challenges and risks, such as inequity, misuse, economic and environmental impact, legal and ethical considerations¹.

Learn more: [1. en.wikipedia.org](#) [2. bing.com](#) [3. research.aimultiple.com](#) [4. research.ibm.com](#)

What are some examples of foundation models?

✓ Searching for: **examples of foundation models**

✓ Generating answers for you...

Some examples of foundation models are large pre-trained language models such as **BERT** and **GPT-3**, which can perform various natural language processing tasks such as email generation, content creation, text summarization, translation, answering questions, and customer support¹². Another example is **DALL-E**, which can generate images from natural language descriptions¹. Foundation models can also be applied to vision, robotics, and other domains³. For example, a financial-services company could customize a foundation model for language to do sentiment analysis⁴.

Learn more: [1. datagen.tech](#) [2. research.aimultiple.com](#) [3. en.wikipedia.org](#) [+2 more](#)

We can ask foundation models what foundation models are by chatting with the new Bing! My questions are in the blue bubbles. Based on my questions, Bing retrieves relevant text from the internet (you can see the search queries in the bold text) so that our results are based off both the rich semantic understanding of the model as well as the relevant information online. Bing also cites its sources.

Foundation models exhibit emergent behavior

As the scale of the model increases, the performance improves across tasks while also **unlocking new capabilities that were not anticipated**. For example, a model trained on a large language dataset might learn to generate stories on its own, or to do arithmetic, without being explicitly programmed to do so.



* <https://crfm.stanford.edu/commentary/2021/10/18/steinhardt.html>

As the size of the model scales up, the capabilities increase, even when these capabilities were not explicitly implemented.
(Gif source - <https://ai.googleblog.com/2022/04/pathways-language-model-palm-scaling-to.html>)

The Evolution of AI in One Year



2022 - 2023



Putting AI to Work

GPT and the Data Professional's Toolbox

Azure OpenAI Service



The graphic displays the Azure OpenAI Service ecosystem. At the top, it lists four models: GPT-3 (GA), DALL-E 2 (preview), ChatGPT (GA), and GPT-4 (GA). Below these are four key capabilities: 'Apply your own data' (Available in Preview early June), 'Plugins for Azure OpenAI Service' (Coming soon), 'Configurable Content Filters' (Available in Preview early June), and 'Provisioned Throughput' (Limited Availability early June). The central message is 'Large pretrained foundation AI models custom-tunable with your parameters and your data'. Below this, it highlights 'Summarization Reasoning over data', 'Writing tools Code generation', and 'ChatGPT The Era of Copilots'. At the bottom, it lists the models again and states 'Foundation of enterprise security, privacy and compliance'.

- <https://learn.microsoft.com/en-us/azure/ai-services/openai/>

Toolbox - <https://www.oneusefulting.org/p/what-ai-can-do-with-a-toolbox-getting>

Searchable Knowledge Basis:

Main Combo: Azure OpenAI + Azure Cognitive Search (<https://lnkd.in/eAhxqjYq>) - Elevate knowledge retrieval via Semantic, Vector, or Hybrid Search.

Alternatives: Azure Data Explorer or Azure Cosmos DB for Mongo DB vCore.

Bonus: Need to handle intricate documents? Integrate Azure AI Document Intelligence (formerly Azure Form Recognizer): <https://lnkd.in/e9cSx5pi>. Looking for a quick win? Consider our accelerator: <https://lnkd.in/eapnZA7p>.

Conversational Interfaces:

Main Combo: Azure OpenAI + Power Virtual Agent - An instant setup with customization flexibility for chat experiences and conversation management.

Alternative: Launch a standard GUI via "Azure OpenAI on Your Data" (<https://lnkd.in/eeXtkT9b>) or explore our GUI-equipped accelerators: <https://lnkd.in/e2ebpjBX>. Enhance conversations with Prompt Flow in Azure ML and utilize the Retrieval Augmentation Pattern (RAG).

Advanced Audio Analysis:

Main Combo: Azure OpenAI + Azure Speech Service (<https://lnkd.in/ewuyVv4W>) - Ideal for summarizing calls or pinpointing call-center insights.

Preprocessing Sensitive Data:

Additional protection: In Azure OpenAI your data remains yours. Yet, some might decide to filter or mask sensitive information to ensure regulatory compliance.

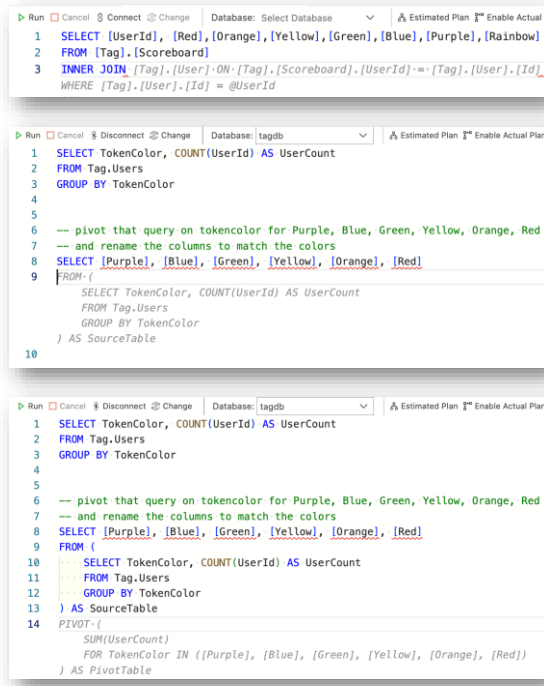
Main Combo: Combine Azure OpenAI and PII Detection in Azure AI Language Service (<https://lnkd.in/ee87F2MJ>) or implement the PII detection skill in Azure Cognitive Search: <https://lnkd.in/edEycSkX>.

Alternative: Presidio SDK (<https://lnkd.in/emNUVnBp>).

Image Interpretation:

Main Combo: Azure OpenAI + Azure Computer Vision (<https://lnkd.in/eVRdVhwZ>) lets GPT models reason over image content, for example, by using object detection, or tagging.

Future & Alternatives: Consider training a personalized vision model or await the arrival of multi-modal capabilities.



```
1 SELECT [UserId], [Red],[Orange],[Yellow],[Green],[Blue],[Purple],[Rainbow]
2 FROM [Tag].[Scoreboard]
3 INNER JOIN [Tag].[User] ON [Tag].[Scoreboard].[UserId]=.[Tag].[User].[Id]
WHERE [Tag].[User].[Id] = @UserId

1 SELECT TokenColor, COUNT(UserId) AS UserCount
2 FROM Tag.Users
3 GROUP BY TokenColor
4
5
6 -- pivot that query on tokencolor for Purple, Blue, Green, Yellow, Orange, Red
7 -- and rename the columns to match the colors
8 SELECT [Purple], [Blue], [Green], [Yellow], [Orange], [Red]
9 FROM (
10     SELECT TokenColor, COUNT(UserId) AS UserCount
11     FROM Tag.Users
12     GROUP BY TokenColor
13 ) AS SourceTable
14
```

```
1 SELECT TokenColor, COUNT(UserId) AS UserCount
2 FROM Tag.Users
3 GROUP BY TokenColor
4
5
6 -- pivot that query on tokencolor for Purple, Blue, Green, Yellow, Orange, Red
7 -- and rename the columns to match the colors
8 SELECT [Purple], [Blue], [Green], [Yellow], [Orange], [Red]
9 FROM (
10     SELECT TokenColor, COUNT(UserId) AS UserCount
11     FROM Tag.Users
12     GROUP BY TokenColor
13 ) AS SourceTable
14 PIVOT (
15     SUM(UserCount)
16     FOR TokenColor IN ([Purple], [Blue], [Green], [Yellow], [Orange], [Red])
17 ) AS PivotTable
```

- <https://learn.microsoft.com/en-us/training/modules/introduction-to-github-copilot/>

References:

- <https://learn.microsoft.com/en-us/sql/azure-data-studio/extensions/github-copilot-extension?view=sql-server-ver16>
- <https://techcommunity.microsoft.com/t5/azure-sql-blog/unleashing-sql-sorcery-increasing-performance-and-complexity/ba-p/3898909>
- <https://techcommunity.microsoft.com/t5/azure-sql-blog/github-copilot-for-sql-developers-turbocharge-your-sql/ba-p/3875915>
- <https://github.com/features/copilot>
- <https://code.visualstudio.com/docs/editor/artificial-intelligence>



Database Integration for Machine Learning and Azure OpenAI

Database as Source, Database as Engine



SQL Server Machine Learning Services

- Support for major languages
- T-SQL for data work
- SSIS for Data Engineering
- Ground to Cloud
- Complete Ecostructure
- Training and Deployment target – Data Stays in a Secure Environment
- Uses the Machine Learning Extensibility Framework



Docs Reference - <https://learn.microsoft.com/en-us/sql/machine-learning/sql-server-machine-learning-services?view=sql-server-ver16>



Using the Extensibility Framework

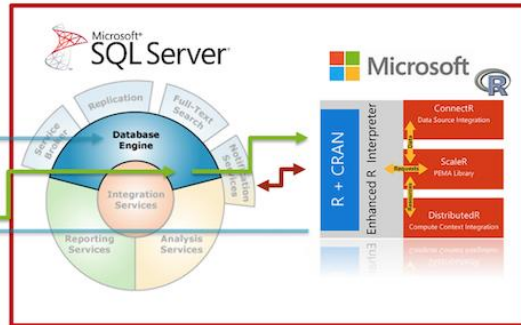
T-SQL and R Interaction

```
EXEC sp_execute_external_script
@language =N'R',

-- SQL Part (sends to @script)
@input_data_1 =N 'SELECT 1 as Installed',

-- R Part (gets @input_data_1)
@script=N'OutputDataSet<-InputDataSet'

WITH RESULT SETS
(([[Installed] int not null));
GO
```



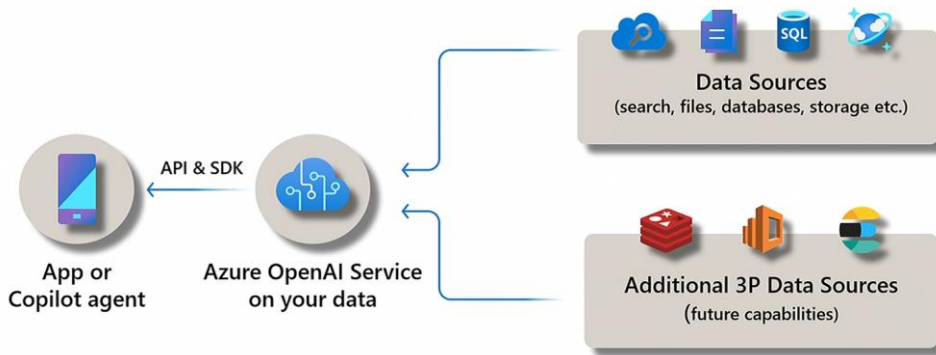


SQL Server ML and OpenAI

GPT Possibilities in SQL Server



Azure OpenAI Service on *your* data (Preview)



* <https://learn.microsoft.com/en-us/azure/ai-services/openai/concepts/use-your-data>



SQL Server ML and OpenAI

The screenshot shows a code editor interface with a file explorer on the left and a code preview on the right. The file explorer shows a directory structure with files like `ai_ml_dl`, `DS_Store`, `AI.ipynb`, `DL.ipynb`, `ML.ipynb`, `OpenAI-SQLML.ipynb`, `README.md`, `ai_whatousewhen`, `bdc`, `dataliteracy`, `graphics`, `presentation_workshop`, `problemsolution`, `secureai`, `shut_the_front_door`, `tdsp`, `DS_Store`, and `LICENSE`. The code preview shows the following content:

```
Preview Code Blame 532 Lines (532 loc) · 27.6 KB
```

Using Microsoft Azure OpenAI and SQL Server Machine Learning Service

Proof-of-Concept Demonstration Project

Buck Woody, Principal Applied Data Scientist, Microsoft - Last edited 06/07/2022

These sample scripts are provided AS IS without warranty of any kind. Microsoft further disclaims all implied warranties including, without limitation, any implied warranties of merchantability or of fitness for a particular purpose. The entire risk arising out of the use or performance of the sample scripts and documentation remains with you.

[\(Business Scenarios for AdventureWorks are here.\)](#)

Restore the AdventureWorks sample database suitable for your installation version. This example uses SQL Server 2019 Machine Learning Services in a Container, where the AdventureWorks backup file has been copied to the Container.

```
In [26]: USE master;
GO
RESTORE DATABASE [AdventureWorks]
FROM DISK = '/var/opt/mssql/data/AdventureWorks2019.bak'
WITH MOVE 'AdventureWorks2019' TO '/var/opt/mssql/data/AdventureWorks_Data.adf'
, MOVE 'AdventureWorks2019_Log' TO '/var/opt/mssql/log/AdventureWorks_Log.ldf'
, RECOVERY, REPLACE, STATS = 10;
```

Commands completed successfully.
Total execution time: 00:00:00.002

Check to ensure Machine Learning Services are installed, turn on if not configured. Also ensure the `SQLLaunchpad` Service is enabled and started in SQL Server Configuration Manager.

You can learn more about [installing SQL Server Machine Learning Services here.](#)

aka.ms/sqlopenai

* <https://cloudblogs.microsoft.com/sqlserver/2023/06/14/secure-your-ai-using-sql-server-machine-learning-with-microsoft-azure-openai-services/>

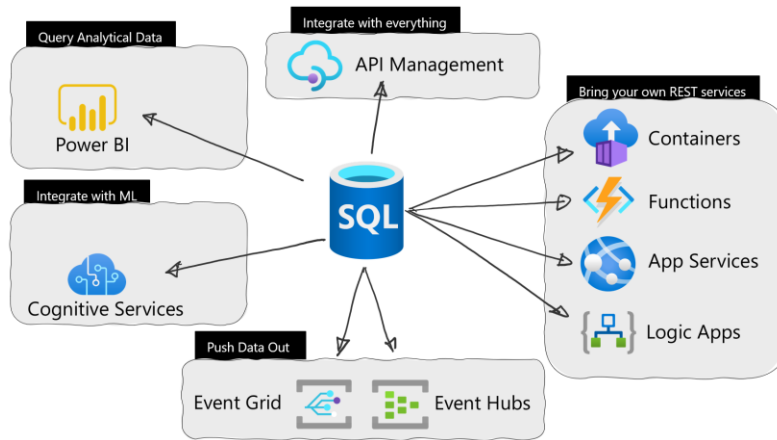


Azure SQL DB and OpenAI

GPT Possibilities in Azure Data



Azure SQL DB and OpenAI



- <https://github.com/JetterMcTedder/azure-sql-db-developers-workshop/blob/main/docs/6-invoke-REST.md>

<https://devblogs.microsoft.com/azure-sql/rest-endpoint-invocation-is-now-generally-available/>

<https://devblogs.microsoft.com/azure-sql/building-your-own-db-copilot-for-azure-sql-with-azure-openai-gpt-4/>

<https://techcommunity.microsoft.com/t5/analytics-on-azure-blog/revolutionizing-sql-queries-with-azure-open-ai-and-semantic/ba-p/3913513>



Microsoft AI

Resources

<https://www.microsoft.com/en-us/ai>

<https://platform.openai.com/docs/guides/fine-tuning>



Thank you!

References:

Comprehensive explanation: <https://mark-riedl.medium.com/a-very-gentle-introduction-to-large-language-models-without-the-hype-5f67941fa59e>

https://python.langchain.com/en/latest/getting_started/tutorials.html

<https://www.kdnuggets.com/2023/05/free-chatgpt-course-openai-api-code-5-projects.html>

<https://news.microsoft.com/source/features/ai/the-art-of-the-prompt-how-to-get-the-best-out-of-generative-ai/>

<https://github.com/Azure-Samples/azure-search-openai-demo/>

<https://blog.seekwell.io/gpt3>

<https://blog.devgenius.io/query-database-using-natural-language-openai-gpt-3-d2403636527a>

<https://learn.microsoft.com/en-us/azure/cognitive-services/openai/concepts/advanced-prompt-engineering?pivots=programming-language-completions>

<https://blog.fabric.microsoft.com/en-us/blog/introducing-synapse-data-science-in-microsoft-fabric/>

<https://dzone.com/articles/introduction-machine-learning-2>

<https://dzone.com/articles/text-classification-with-bert>

<https://dzone.com/articles/how-i-turned-my-companys-docs-into-a-searchable-da>

<https://t.co/wxrBtHCmlC> <https://t.co/nCjdA7hj2Z>

<https://github.com/jconorgrogan/CLARKGPT>

<https://medium.com/@dpralay07/deploy-a-python-flask-application-in-iis-server-and-run-on-machine-ip-address-ddb81df8edf3>

<https://dzone.com/articles/how-to-create-question-answering-model-from-scratc>

<https://www.sqlservercentral.com/articles/how-to-generate-images-with-ai-and-store-them-in-sql-server-using-python-and-dall%C2%B7e>