What is Gen AI ? Explain with example

Generative AI refers to a class of artificial intelligence models that can create new content, such as text, images, audio, or even code, based on the patterns it has learned from a given dataset. Instead of just analyzing or categorizing data, generative AI models generate new outputs that resemble the data they were trained on.

How It Works:

Generative AI models are typically based on neural networks, particularly deep learning techniques, such as:

1. Generative Adversarial Networks (GANs)
2. Variational Autoencoders (VAEs)
3. Transformers based models
4. Auto Regressive Models
5. Diffusion Models
6. Neural Radiance Fields
7. Neural Style Transfer

Structure of GANs:

A GAN consists of two neural networks that compete against each other:

1. **Generator (G)**: This network takes in random noise (a latent space) and tries to generate realistic-looking data (e.g., images) based on the patterns it learned from the dataset.
2. **Discriminator (D)**: This network evaluates the data, deciding whether a given sample is real (from the actual dataset) or fake (generated by the Generator).

How GANs Work:

The training process is like a game between the Generator and the Discriminator:

1. **Generator's Role**: The Generator takes random noise as input and produces data (e.g., an image). In the beginning, this generated data is random and unrealistic.
2. **Discriminator's Role**: The Discriminator is shown both real data (from the training set) and fake data (generated by the Generator). It tries to classify each piece of data as either real or fake.
3. **Backpropagation**: The Discriminator's feedback is used to adjust both networks:
   - The Discriminator improves by learning to better classify real and fake data.
   - The Generator improves by learning to create more convincing data that can "fool" the Discriminator.

This back-and-forth competition continues until the Generator becomes so good that the Discriminator can no longer reliably distinguish between real and generated data.

Applications of GANs:

1. **Image Generation**: GANs can generate realistic images from random noise, such as human faces, landscapes, and objects.
   - Example: **This Person Does Not Exist** is a website that uses GANs to generate random, realistic-looking human faces.
2. **Image-to-Image Translation**: GANs can transform one image into another. For example, converting sketches into detailed images or black-and-white images into color.
   - Example: **Pix2Pix** converts hand-drawn sketches into photorealistic images.
3. **Text-to-Image Generation**: GANs can generate images from text descriptions, as seen in models like **DALL-E**.

4. **Super-Resolution**: GANs can enhance the resolution of images, making them sharper and more detailed.
    - Example: **ESRGAN (Enhanced Super-Resolution GAN)** can upscale low-resolution images to high resolution.
5. **Deepfake Generation**: GANs are used in generating deepfakes, which involve replacing a person's face in a video or image with someone else's.

LLMs

An LLM stands for Large Language Model. It refers to a type of artificial intelligence (AI) model designed to understand, generate, and process human language at a large scale. These models are trained on vast amounts of text data to perform a variety of tasks, such as:

- Text generation (e.g., writing essays, stories, or articles)
- Translation between languages
- Answering questions (like what I'm doing now)
- Summarizing text
- Analyzing sentiment
- Code generation

The most well-known LLMs include models like GPT (Generative Pre-trained Transformer), BERT (Bidirectional Encoder Representations from Transformers), and T5 (Text-to-Text Transfer Transformer). LLMs use deep learning, typically with a neural network architecture known as a transformer, which allows them to capture the context and relationships between words in a sophisticated way.

Embeddings in NLP

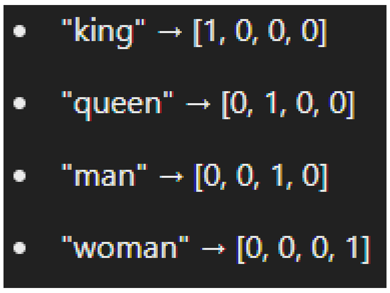
Embeddings are a way to represent data, typically textual or categorical information, as numerical vectors in a continuous vector space. The goal of embeddings is to capture the semantic meaning or relationships between pieces of data in a way that similar items are close together in this vector space, while dissimilar ones are far apart.

In natural language processing (NLP), word embeddings are commonly used to convert words or phrases into fixed-size vectors of real numbers, enabling machine learning models to process and learn from textual data more effectively.

Example of Embeddings in NLP:

Let’s consider the following words: "king," "queen," "man," and "woman."

A simple way to represent these words might be to use one-hot encoding, where each word is represented as a unique binary vector. For example:

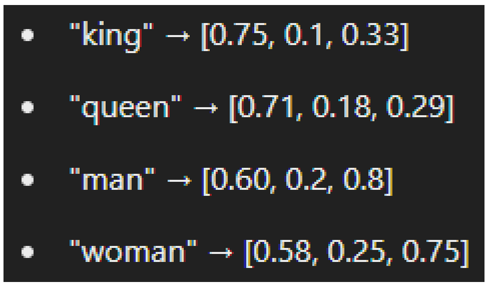


- "king" → [1, 0, 0, 0]
- "queen" → [0, 1, 0, 0]
- "man" → [0, 0, 1, 0]
- "woman" → [0, 0, 0, 1]

The issue with this approach is that it doesn't capture any relationship between the words. For instance, "king" and "queen" are related in terms of royalty, and "man" and "woman" are related in terms of gender, but this is not reflected in one-hot encoding.

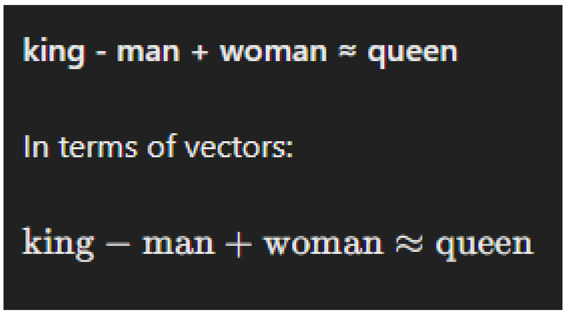
Word Embeddings Example:

Embeddings solve this problem by placing words in a multi-dimensional space where similar words are closer together. After training on a large dataset, the embeddings might look like this in a 3-dimensional space:



- "king" → [0.75, 0.1, 0.33]
- "queen" → [0.71, 0.18, 0.29]
- "man" → [0.60, 0.2, 0.8]
- "woman" → [0.58, 0.25, 0.75]

Here, the relationships between words can be captured in terms of their vector distances. One famous example of the power of embeddings is the analogy:



king - man + woman ≈ queen

In terms of vectors:

$$\text{king} - \text{man} + \text{woman} \approx \text{queen}$$

This shows that embeddings are capturing semantic relationships, like the gender difference between "king" and "queen."

Use Case: Sentence Embeddings

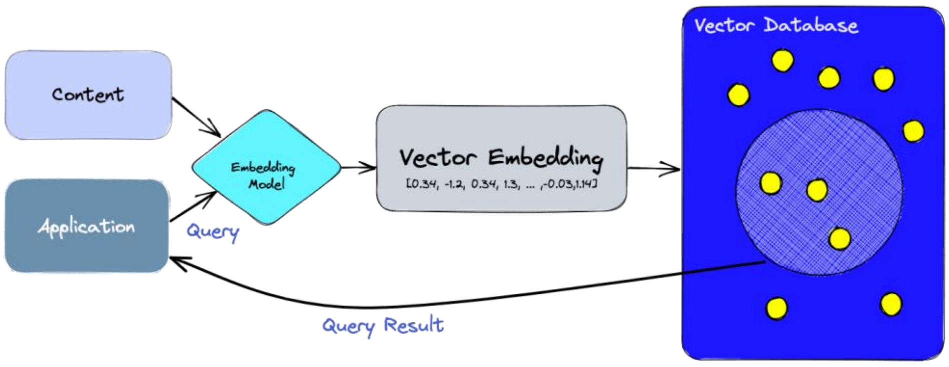
Embeddings can be used not just for words but also for entire sentences. For instance, two sentences that convey the same meaning will have vectors that are close to each other:

- "I am happy." → [0.2, 0.4, 0.7, ...]
- "I feel joyful." → [0.19, 0.41, 0.68, ...]

In Summary:

Embeddings help transform complex data (like words or sentences) into a numerical format that preserves semantic relationships, enabling machine learning models to understand and work with the data more effectively. They're widely used in applications like search engines, machine translation, and recommendation systems.

Vector Databases



https://www.pinecone.io/learn/vector-database/

Retrieval-Augmented Generation (RAG) is a hybrid approach that combines the power of information retrieval and natural language generation. It is primarily used in large language models like GPT to improve the accuracy and relevance of generated responses, especially for tasks that require factual or domain-specific knowledge.

Here's how it works:

1. Retrieval Component: When given a query, the system first retrieves relevant documents or knowledge from a large external dataset, such as a database or collection of text, using an information retrieval system (like a search engine).
2. Augmentation: The retrieved information is then used to augment the input to the language model, providing it with specific context or facts that can help generate more accurate or informed responses.
3. Generation Component: Finally, the language model (such as GPT) uses this augmented information to generate a response. The response is not purely based on the pre-trained model but also integrates the factual data retrieved in the first step.

Transformer-based Models

https://jalammar.github.io/illustrated-transformer/

Innovations in Transformers

1. Positional Encoding
2. Attention
3. Self Attention

Embedding Algorithms in Transformers:

- **Purpose**: In Transformers, data like words or tokens must be converted into numerical representations (vectors) for the model to process. Embedding algorithms are used to map these discrete tokens (like words) into continuous high-dimensional vectors. The goal is to capture semantic information about each token, such that similar words or tokens are represented by vectors that are close together in this space.
- **How It Works**:
  - An embedding layer is a learned lookup table. Each unique token in the input sequence is assigned a vector, and during training, these vectors are adjusted based on the task the model is trying to solve.
  - For example, in a sentence, words like "cat" and "dog" might be closer in the embedding space than "cat" and "car," reflecting their similarity in meaning.
- **Example in NLP**: If you input a word like "king," the embedding layer will produce a dense vector (e.g., a 512-dimensional vector) that represents the meaning of "king" in relation to other words in the dataset.
- **Common Embedding Techniques**:
  - **Word2Vec** and **GloVe**: Pretrained embeddings that map words into vectors based on their co-occurrence in large datasets.

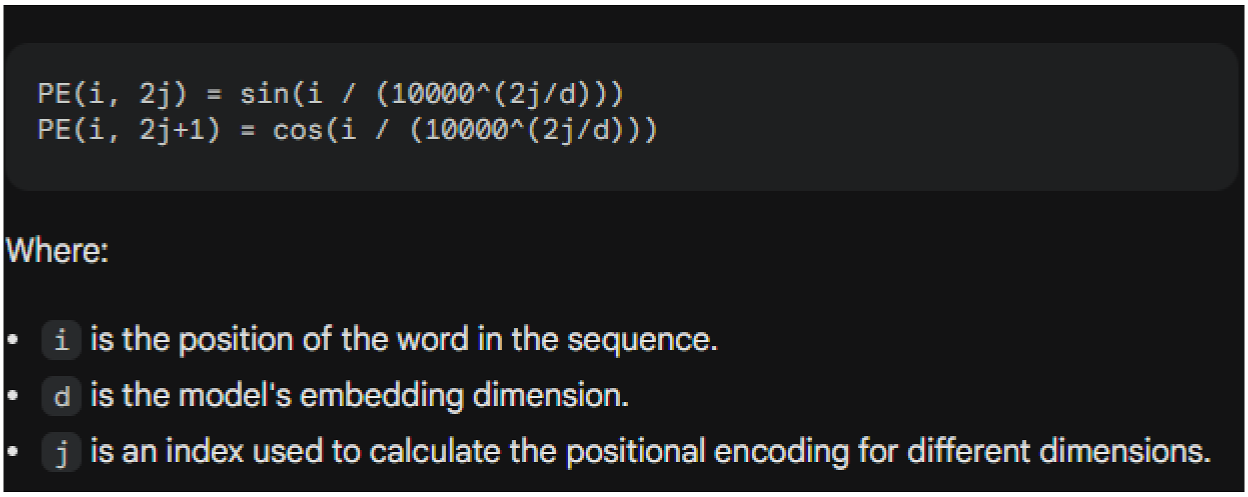
Positional Encoding – A Simple Example

In Transformer models, **positional encoding** helps the model understand the order of words or tokens in a sequence since the self-attention mechanism doesn't inherently include any notion of position.

How Does it Work?

Imagine a sequence of words: "The quick brown fox jumps over the lazy dog." Each word is represented as a dense vector embedding. Without positional encoding, the model would treat each word independently, unaware of its position in the sequence.

Positional encoding adds a positional vector to each word embedding. This positional vector is calculated based on the position of the word in the sequence. The formula for calculating the positional encoding vector for a position i is as follows:



```
PE(i, 2j) = sin(i / (10000^(2j/d)))
PE(i, 2j+1) = cos(i / (10000^(2j/d)))
```

Where:

- `i` is the position of the word in the sequence.
- `d` is the model's embedding dimension.
- `j` is an index used to calculate the positional encoding for different dimensions.

Example:

Consider a sequence of three words: "cat", "dog", and "bird". Let's assume the embedding dimension is 4.

- **Position 1 (cat):**
  - PE(1, 0) = sin(1 / (10000^(0/4)))
  - PE(1, 1) = cos(1 / (10000^(0/4)))
  - PE(1, 2) = sin(1 / (10000^(2/4)))
  - PE(1, 3) = cos(1 / (10000^(2/4)))
- **Position 2 (dog):**
  - ...
- **Position 3 (bird):**
  - ...

The calculated positional encoding vectors are added to the word embeddings, providing the model with information about the position of each word in the sequence.

By incorporating positional encoding, transformer models can effectively capture the sequential nature of data, enabling them to perform well on tasks like machine translation, text summarization, and question answering.