Generative AI & Large Language Models Glossary

Version 1

Gathered by Faisal Alsrheed
https://www.linkedin.com/in/faisalalsrheed/

Term	**Definition**	**Reference /Source**
A foundation model	“A foundation model is a large artificial intelligence model trained on a vast quantity of unlabeled data at scale (usually by self-supervised learning) resulting in a model that can be adapted to a wide range of downstream tasks.Foundation models have helped bring about a major transformation in how AI systems are built since their introduction in 2018. Early examples of foundation models were large pre-trained language models including BERT and GPT-3.”	https://en.wikipedia.org/wiki/Foundation_models
GPT (Generative Pre-trained Transformer)	“A family of Transformer-based large language models developed by OpenAI.” GPT variants can apply to multiple modalities, including: image generation (for example, ImageGPT) text-to-image generation (for example, DALL-E).	https://developers.google.com/machine-learning/glossary#GPT
pre-trained model	“Models or model components (such as embedding vector) that have been already been trained. Sometimes, you'll feed pre-trained embedding vectors into a neural network. Other times, your model will train the embedding vectors itself rather than rely on the pre-trained embeddings.”	https://developers.google.com/machine-learning/glossary#pre-trained-model
Embedding vector	“Broadly speaking, an array of floating-point numbers taken from any hidden layer that describe the inputs to that hidden layer. Often, an embedding vector is the array of floating-point numbers trained in an embedding layer.” “An embedding vector is not a bunch of random numbers. An embedding layer determines these values through training, similar to the way a neural network learns other weights during training. Each element of the array is a rating along some characteristic of a tree species. Which element represents which tree species' characteristic? That's very hard for humans to determine. The mathematically remarkable part of an embedding vector is that similar items have similar sets of floating-point numbers. For example, similar tree species have a more similar set of floating-point numbers than dissimilar tree species. Redwoods and sequoias are related tree species, so they'll have a more similar set of floating-pointing numbers than redwoods and coconut palms. The numbers in the embedding vector will change each time you retrain the	https://developers.google.com/machine-learning/glossary#embedding_vector

	model, even if you retrain the model with identical input.”	
word embedding	“Representing each word in a word set within an embedding vector; that is, representing each word as a vector of floating-point values between 0.0 and 1.0. Words with similar meanings have more-similar representations than words with different meanings.”	https://developers.google.com/machine-learning/glossary#word-embedding
embedding layer	“A special hidden layer that trains on a high-dimensional categorical feature to gradually learn a lower dimension embedding vector. An embedding layer enables a neural network to train far more efficiently than training just on the high-dimensional categorical feature.”	https://developers.google.com/machine-learning/glossary#embedding_layer
embedding space	“The d-dimensional vector space that features from a higher-dimensional vector space are mapped to. Ideally, the embedding space contains a structure that yields meaningful mathematical results; for example, in an ideal embedding space, addition and subtraction of embeddings can solve word analogy tasks. The dot product of two embeddings is a measure of their similarity.”	https://developers.google.com/machine-learning/glossary#embedding-space
token	“In a language model, the atomic unit that the model is training on and making predictions on. A token is typically one of the following: a word—for example, the phrase "dogs like cats" consists of three word tokens: "dogs", "like", and "cats". a character—for example, the phrase "bike fish" consists of nine character tokens. (Note that the blank space counts as one of the tokens.) subwords—in which a single word can be a single token or multiple tokens. A subword consists of a root word, a prefix, or a suffix. For example, a language model that uses subwords as tokens might view the word "dogs" as two tokens (the root word "dog" and the plural suffix "s"). That same language model might view the single word "taller" as two subwords (the root word "tall" and the suffix "er").	https://developers.google.com/machine-learning/glossary#token

	.”	
modality	A high-level data category. For example, numbers, text, images, video, and audio are five different modalities.	https://developers.google.com/machine-learning/glossary#modality
multimodal model	“A model whose inputs and/or outputs include more than one modality. For example, consider a model that takes both an image and a text caption (two modalities) as features, and outputs a score indicating how appropriate the text caption is for the image. So, this model's inputs are multimodal and the output is unimodal.”	https://developers.google.com/machine-learning/glossary#multimodal-model
Transformer	A neural network architecture developed at Google that relies on self-attention mechanisms to transform a sequence of input embeddings into a sequence of output embeddings without relying on convolutions or recurrent neural networks. A Transformer can be viewed as a stack of self-attention layers. “A Transformer can include any of the following: ● an encoder ● a decoder ● both an encoder and decoder An encoder transforms a sequence of embeddings into a new sequence of the same length. An encoder includes N identical layers, each of which contains two sub-layers. These two sub-layers are applied at each position of the input embedding sequence, transforming each element of the sequence into a new embedding. The first encoder sub-layer aggregates information from across the input sequence. The second encoder sub-layer transforms the aggregated information into an output embedding. A decoder transforms a sequence of input embeddings into a sequence of output embeddings, possibly with a different length. A decoder also includes N identical layers with three sub-layers, two of which are similar to the encoder sub-layers. The third decoder sub-layer takes the output of the encoder and applies the self-attention mechanism to gather information from it.”	https://developers.google.com/machine-learning/glossary#Transformer
encoder	In general, any ML system that converts from a raw, sparse, or external representation into a more processed, denser, or	https://developers.google.com/machine-learning/gloss

	more internal representation. Encoders are often a component of a larger model, where they are frequently paired with a decoder. Some Transformers pair encoders with decoders, though other Transformers use only the encoder or only the decoder. Some systems use the encoder's output as the input to a classification or regression network.	ary#encoder
decoder	“In general, any ML system that converts from a processed, dense, or internal representation to a more raw, sparse, or external representation. Decoders are often a component of a larger model, where they are frequently paired with an encoder. In sequence-to-sequence tasks, a decoder starts with the internal state generated by the encoder to predict the next sequence.”	https://developers.google.com/machine-learning/glossary#decoder
self-attention layer	“A neural network layer that transforms a sequence of embeddings (for instance, token embeddings) into another sequence of embeddings. Each embedding in the output sequence is constructed by integrating information from the elements of the input sequence through an attention mechanism. The self part of self-attention refers to the sequence attending to itself rather than to some other context. Self-attention is one of the main building blocks for Transformers and uses dictionary lookup terminology, such as “query”, “key”, and “value”.” A self-attention layer starts with a sequence of input representations, one for each word. The input representation for a word can be a simple embedding. For a sequence of *n* tokens, self-attention transforms a sequence of embeddings *n* separate times, once at each position in the sequence.	https://developers.google.com/machine-learning/glossary#self-attention
Attention	“Any of a wide range of neural network architecture mechanisms that aggregate information from a set of inputs in a data-dependent manner. A typical attention mechanism	https://developers.google.com/machine-learning/glossary#attention

	might consist of a weighted sum over a set of inputs, where the weight for each input is computed by another part of the neural network.”	
language model	“A model that estimates the probability of a token or sequence of tokens occurring in a longer sequence of tokens”	https://developers.google.com/machine-learning/glossary#language-model
Large Language Models (LLMs)	“An informal term with no strict definition that usually means a language model that has a high number of parameters. Some large language models contain over 100 billion parameters.”	https://developers.google.com/machine-learning/glossary#large-language-model
Text Generation	“Software that generates coherent human language”	https://txt.cohere.ai/generative-ai-future-or-present/
Large language models	“An informal term with no strict definition that usually means a language model that has a high number of parameters. Some large language models contain over 100 billion parameters.” “You might be wondering when a language model becomes large enough to be termed a large language model. Currently, there is no agreed-upon defining line for the number of parameters.”	https://developers.google.com/machine-learning/glossary#large-language-model
Vector database	“a vector database is a fully managed, no-frills solution for storing, indexing, and searching across a massive dataset of unstructured data that leverages the power of embeddings from machine learning models.” “Vector database options include: ● Pinecone, a fully managed vector database ● Weaviate, an open-source vector search engine ● Redis as a vector database ● Qdrant, a vector search engine ● Milvus, a vector database built for scalable similarity search ● Chroma, an open-source embeddings store”	https://zilliz.com/learn/what-is-vector-database https://platform.openai.com/docs/guides/embeddings/how-can-i-retrieve-k-nearest-embedding-vectors-quickly

retrieval augmented generation	“The idea of retrieval augmented generation is that when given a question you first do a retrieval step to fetch any relevant documents.”	https://docs.langchain.com/docs/use-cases/qa-docs
LangChain	“LangChain is a framework for developing applications powered by language models. We believe that the most powerful and differentiated applications will not only call out to a language model via an api, but will also: **Be data-aware**: connect a language model to other sources of data **Be agentic:** allow a language model to interact with its environment”	https://docs.langchain.com/docs/
LaMDA (Language Model for Dialogue Applications)	“A Transformer-based large language model developed by Google trained on a large dialogue dataset that can generate realistic conversational responses.”	https://developers.google.com/machine-learning/glossary#lamda-language-model-for-dialogue-applications
unidirectional language model	“A language model that bases its probabilities only on the tokens appearing before, not after, the target token(s). Contrast with bidirectional language model.”	https://developers.google.com/machine-learning/glossary#unidirectional-language-model
BERT (Bidirectional Encoder Representations from Transformers)	“A model architecture for text representation. A trained BERT model can act as part of a larger model for text classification or other ML tasks. BERT has the following characteristics: ● Uses the Transformer architecture, and therefore relies on self-attention. ● Uses the encoder part of the Transformer. The encoder's job is to produce good text representations, rather than to perform a specific task like classification. ● Is bidirectional. ● Uses masking for unsupervised training.”	https://developers.google.com/machine-learning/glossary#bert-bidirectional-encoder-representations-from-transformers
GPT-4	● “GPT-4 is a Transformer based model pre-trained to predict the next token in a document” ● GPT-4 is the latest and most powerful model from OpenAI.	https://platform.openai.com/docs/introduction/next-steps

	● “GPT-4 is a large multimodal model (accepting image and text inputs, emitting text outputs) that, while less capable than humans in many real-world scenarios, exhibits human-level performance on various professional and academic benchmarks.” ● “GPT-4 can accept a prompt of text and images, which—parallel to the text-only setting—lets the user specify any vision or language task.” ● “GPT-4 exhibits human-level performance on various professional and academic benchmarks” ● “GPT-4 is a Transformer-style model pre-trained to predict the next token in a document, using both publicly available data (such as internet data) and data licensed from third-party providers. The model was then fine-tuned using Reinforcement Learning from Human Feedback (RLHF)”	https://openai.com/research/gpt-4 https://arxiv.org/abs/2303.08774
ChatGPT	● ChatGPT developed by OpenAI and launched in November 2022. ● “ChatGPT is fine-tuned from a model in the GPT-3.5 series, which finished training in early 2022.” ● “ChatGPT and GPT-3.5 were trained on an Azure AI supercomputing infrastructure.” ● “ChatGPT interacts in a conversational way. The dialogue format makes it possible for ChatGPT to answer followup questions, admit its mistakes, challenge incorrect premises, and reject inappropriate requests.” ● ChatGPT is a sibling model to InstructGPT, which is trained to follow an instruction in a prompt and provide a detailed response. ● GPT-3.5-Turbo is the model that powers ChatGPT. ● “We trained this model using Reinforcement Learning from Human Feedback (RLHF), using the same methods as InstructGPT, but with slight differences in the data collection setup. We trained an initial model using supervised fine-tuning: human AI trainers provided conversations in which they played both sides—the user and an AI assistant. We gave the trainers access to model-written suggestions to help them compose their responses. We mixed this new dialogue dataset with the InstructGPT dataset, which we transformed into a dialogue format.”	https://platform.openai.com/docs/introduction/next-steps https://openai.com/research/learning-from-human-preferences
ChatGPT Plus	“The new subscription plan, ChatGPT Plus, will be available	https://openai.com/blog/chat

	for $20/month, and subscribers will receive a number of benefits: ● General access to ChatGPT, even during peak times ● Faster response times ● Priority access to new features and improvements”	gpt-plus
Reinforcement learning from human feedback (RLHF)	“In machine learning, reinforcement learning from human feedback (RLHF) or reinforcement learning from human preferences is a technique that trains a "reward model" directly from human feedback and uses the model as a reward function to optimize an agent's policy using reinforcement learning (RL) through an optimization algorithm like Proximal Policy Optimization”	https://en.wikipedia.org/wiki/Reinforcement_learning_from_human_feedback
Chat Plugins	“OpenAI plugins connect ChatGPT to third-party applications. These plugins enable ChatGPT to interact with APIs defined by developers, enhancing ChatGPT's capabilities and allowing it to perform a wide range of actions. Plugins can allow ChatGPT to do things like: ● Retrieve real-time information; e.g., sports scores, stock prices, the latest news, etc. ● Retrieve knowledge-base information; e.g., company docs, personal notes, etc. ● Perform actions on behalf of the user; e.g., booking a flight, ordering food, etc.	
GPT Fine-tuning	“Fine-tuning lets you get more out of the models available through the API by providing: ● Higher quality results than prompt design ● Ability to train on more examples than can fit in a prompt ● Token savings due to shorter prompts ● Lower latency requests GPT-3 has been pre-trained on a vast amount of text from the open internet. When given a prompt with just a few examples, it can often intuit what task you are trying to perform and generate a plausible completion. This is often called "few-shot learning." Fine-tuning improves on few-shot learning by training on many more examples than can fit in the prompt, letting you	https://platform.openai.com/docs/guides/fine-tuning

	achieve better results on a wide number of tasks. Once a model has been fine-tuned, you won't need to provide examples in the prompt anymore. This saves costs and enables lower-latency requests. At a high level, fine-tuning involves the following steps: ● Prepare and upload training data ● Train a new fine-tuned model ● Use your fine-tuned model"	
embedding	"An embedding is a vector (list) of floating point numbers. The distance between two vectors measures their relatedness. Small distances suggest high relatedness and large distances suggest low relatedness."	https://platform.openai.com/docs/guides/embeddings/what-are-embeddings
OpenAI's text embeddings	"OpenAI's text embeddings measure the relatedness of text strings. Embeddings are commonly used for: ● Search (where results are ranked by relevance to a query string) ● Clustering (where text strings are grouped by similarity) ● Recommendations (where items with related text strings are recommended) ● Anomaly detection (where outliers with little relatedness are identified) ● Diversity measurement (where similarity distributions are analyzed) ● Classification (where text strings are classified by their most similar label)"	https://platform.openai.com/docs/guides/embeddings/what-are-embeddings
text-embedding-ada-002	● New and improved embedding model from OpenAI. ● "text-embedding-ada-002 outperforms all the old embedding models on text search, code search, and sentence similarity tasks and gets comparable performance on text classification."	https://openai.com/blog/new-and-improved-embedding-model
OpenAI Codex	● "AI system that translates natural language to code. ● "OpenAI Codex is a general-purpose programming model, meaning that it can be applied to essentially any programming task" ● "Codex is the model that powers GitHub Copilot, which we built and launched in partnership with GitHub." ● "OpenAI Codex is a descendant of GPT-3; its training	https://openai.com/blog/openai-codex https://github.com/features/copilot/

	data contains both natural language and billions of lines of source code from publicly available sources, including code in public GitHub repositories. OpenAI Codex is most capable in Python, but it is also proficient in over a dozen languages” ● “Trained on billions of lines of code”	
GitHub Copilot	● “GitHub Copilot is a cloud-based artificial intelligence tool developed by GitHub and OpenAI to assist users of Visual Studio Code, Visual Studio, Neovim, and JetBrains integrated development environments (IDEs) by autocompleting code.Currently available by subscription to individual developers, the tool was first announced by GitHub on 29 June 2021, and works best for users coding in Python, JavaScript, TypeScript, Ruby, and Go.” ● GitHub Copilot uses the OpenAI Codex to suggest code and entire functions in real-time, right from your editor.	https://en.wikipedia.org/wiki/GitHub_Copilot https://github.com/features/copilot/
GitHub Copilot X	● “Copilot X” vision, that includes a new ChatGPT-like experience inside code editors, allowing the chatbot to recognize and explain code and recommend changes and fix bugs.” ● “With chat and terminal interfaces, support for pull requests, and early adoption of OpenAI’s GPT-4, GitHub Copilot X is our vision for the future of AI-powered software development. Integrated into every part of your workflow.” ● It can help with Automate automated testing. ● Copilot X gets chat and voice support.	https://github.com/features/preview/copilot-x https://www.youtube.com/watch?v=5XheKKZoGnE
Azure OpenAI Service	“Azure OpenAI Service provides REST API access to OpenAI's powerful language models including the GPT-3, Codex and Embeddings model series. These models can be easily adapted to your specific task including but not limited to content generation, summarization, semantic search, and natural language to code translation. Users can access the service through REST APIs, Python SDK, or our web-based interface in the Azure OpenAI Studio.”	https://learn.microsoft.com/en-us/azure/cognitive-services/openai/overview https://azure.microsoft.com/en-us/blog/chatgpt-is-now-available-in-azure-openai-service/

Prompt engineering	“Prompt engineering is a concept in artificial intelligence (AI), particularly natural language processing (NLP). In prompt engineering, the description of the task is embedded in the input, e.g., as a question instead of it being implicitly given. Prompt engineering typically works by converting one or more tasks to a prompt-based dataset and training a language model with what has been called "prompt-based learning" or just "prompt learning". Prompt engineering may work from a large "frozen" pretrained language model and where only the representation of the prompt is learned (i.e., optimized), using methods such as "prefix-tuning" or "prompt tuning".”	https://en.wikipedia.org/wiki/Prompt_engineering
Chain-of-thought prompting	“Chain-of-thought (CoT) prompting is a technique for improving the reasoning ability of large language models by prompting them to generate a series of intermediate steps that lead to the final answer of a multi-step problem. It was first proposed by Google researchers in 2022.”	https://en.wikipedia.org/wiki/Chain-of-thought_prompting
few-shot prompting	“In natural language processing, few-shot learning or few-shot prompting is a prompting technique that allows a model to process examples before attempting a task” “Few-shot learning was initially proposed as an alternative to fine-tuning a pre-trained language model on a task-specific dataset. The main advantages of few-shot learning over fine-tuning are a reduction in the amount of task-specific data needed and a reduced potential of overfitting by learning an overly narrow distribution from a large but narrow fine-tuning dataset”	https://en.wikipedia.org/wiki/Few-shot_learning_(natural_language_processing)

Appendix A: List of large language models

Name	Release date [a]	Developer	Number of parameters[b]	Corpus size	License[c]	Notes
BERT	2018	Google	340 million[15]	3.3 billion words[15]	Apache 2.0[16]	early and influential language model[1]
GPT-2	2019	OpenAI	1.5 billion[17]	40GB[18] (~10 billion tokens)[19]	MIT[20]	general-purpose model based on transformer architecture
GPT-3	2020	OpenAI	175 billion[8]	499 billion tokens[19]	public web API	A fine-tuned variant of GPT-3, termed GPT-3.5, was made available to the public through a web interface called ChatGPT in 2022.[21]
GPT-Neo	March 2021	EleutherAI	2.7 billion[22]	825 GiB[23]	MIT[24]	The first of a series of free GPT-3 alternatives released by EleutherAI. GPT-Neo outperformed an equivalent-size GPT-3 model on some benchmarks, but was significantly worse

						than the largest GPT-3.[24]
GPT-J	June 2021	EleutherAI	6 billion[25]	825 GiB[23]	Apache 2.0	GPT-3-style language model
Megatron-Turing NLG	October 2021[26]	Microsoft and Nvidia	530 billion[27]	338.6 billion tokens[27]	Restricted web access	Standard architecture but trained on a supercomputing cluster.
Ernie 3.0 Titan	December 2021	Baidu	260 billion[28][29]	4 Tb	Proprietary	Chinese-language LLM. Ernie Bot is based on this model.
Claude[30]	December 2021	Anthropic	52 billion[31]	400 billion tokens[31]	Closed beta	Fine-tuned for desirable behavior in conversations.[32]
GLaM (Generalist Language Model)	December 2021	Google	1.2 trillion[33]	1.6 trillion tokens[33]	Proprietary	Sparse mixture-of-experts model, making it more expensive to train but cheaper to run inference compared to GPT-3.
Gopher	December 2021	DeepMind	280 billion[34]	300 billion tokens[35]	Proprietary	
LaMDA (Language Models for Dialog Applications)	January 2022	Google	137 billion[36]	1.56T words,[36] 168 billion tokens[35]	Proprietary	Specialized for response generation in conversations. Used in Google Bard chatbot.

GPT-NeoX	February 2022	EleutherAI	20 billion[37]	825 GiB[23]	Apache 2.0	based on the Megatron architecture
Chinchilla	March 2022	DeepMind	70 billion[38]	1.4 trillion tokens[38][35]	Proprietary	Reduced-parameter model trained on more data. Used in the Sparrow bot.
PaLM (Pathways Language Model)	April 2022	Google	540 billion[39]	768 billion tokens[38]	Proprietary	aimed to reach the practical limits of model scale
OPT (Open Pretrained Transformer)	May 2022	Meta	175 billion[40]	180 billion tokens[41]	Non-commercial research[d]	GPT-3 architecture with some adaptations from Megatron
YaLM 100B	June 2022	Yandex	100 billion[42]	1.7TB[42]	Apache 2.0	English-Russian model based on Microsoft's Megatron-LM.
Minerva	June 2022	Google	540 billion[43]	38.5B tokens from webpages filtered for mathematical content and from papers submitted to the arXiv preprint server[43]	Proprietary	LLM trained for solving "mathematical and scientific questions using step-by-step reasoning".[44] Minerva is based on PaLM model, further trained on mathematical and scientific data.
BLOOM	July 2022	Large collaboration led by	175 billion[9]	350 billion tokens (1.6TB)[45]	Responsible AI	Essentially GPT-3 but trained on a multi-lingual corpus (30%

		Hugging Face				English excluding programming languages)
AlexaTM (Teacher Models)	November 2022	Amazon	20 billion[46]	1.3 trillion[47]	public web API[48]	bidirectional sequence-to-sequence architecture
LLaMA (Large Language Model Meta AI)	February 2023	Meta	65 billion[49]	1.4 trillion[49]	Non-commercial research[e]	Trained on a large 20-language corpus to aim for better performance with fewer parameters.[49] Researchers from Stanford University trained a fine-tuned model based on leaked LLaMA weights, called Alpaca.[50]
GPT-4	March 2023	OpenAI	Unknown[f]	Unknown	public web API	Available for ChatGPT Plus users and used in several products.
Cerebras-GPT	March 2023	Cerebras	13 billion[52]		Apache 2.0	Trained with Chinchilla formula

Reference /Source
https://en.wikipedia.org/wiki/Large_language_model