Generative AI and Large Language Models

Prof. K. R. Chowdhary

Former Prof. & Head
Department of Computer Science & Engineering
MBM Engineering College
Web: https://krchowdhary.com/

2026-03-28

Artificial Intelligence

Artificial Intelligence (AI) aims to build systems that perform tasks requiring human intelligence.

- Learning from data
- Pattern recognition
- Natural language understanding
- Decision making

Examples: speech recognition, recommendation systems, machine translation.

Evolution of AI

Major phases:

- Rule-based AI
- Machine learning
- Deep learning
- Generative AI and foundation models

Recent progress is driven by:

- large datasets
- powerful GPUs
- improved neural architectures

Generative AI

Generative AI (GenAI) models learn data distributions and generate new samples.

$$x \sim p_{\text{data}}(x)$$

Goal:

$$\text{Learn } p_\theta(x) \approx p_{\text{data}}(x)$$

Applications:

- Text generation
- Image synthesis
- Code generation
- Speech generation

Discriminative Models

Discriminative models learn the mapping

$$x \rightarrow y$$

They estimate

$$P(y|x)$$

Examples:

- image classification
- spam detection
- sentiment analysis

Generative Models

Generative models learn the data distribution.

$$P(x)$$

or

$$P(x, y)$$

They can generate new realistic samples.
Examples:

- language models
- GANs
- diffusion models

Language Models: $p(x_1, x_2, ..., x_T)$

Language Modeling

A language model assigns probability to a sequence of words.
For sequence $x_1, \ldots, x_T$:

$$P(x_1, x_2, \ldots, x_T) = \prod_{t=1}^{T} P(x_t | x_1, \ldots, x_{t-1})$$

Goal: predict the next word.
Example:
"The capital of India is $\rightarrow$ New Delhi"

Training Objective

Language models are trained by minimizing *cross-entropy* loss.

$$L = -\sum_{t=1}^{T} \log P(x_t | x_{<t})$$

If the correct word receives high probability, the loss becomes small. (Prob of a city as capital of India?

Case 1: Model is confident

$P(NewDelhi|capital) = 0.9$, $\log(0.9) = -0.105$

$-(-0.105) = 0.105$ (This is small loss)

Case 2: Model is very wrong

$P(Daultabaad|capital) = 0.01$

$-\log(0.01) \approx 4.60$ (This is very large loss.)

Optimization is performed using gradient descent.

Recurrent Neural Networks (RNNs)

Motivation: Model the sequential or time-series data.
Examples:

- Natural language sentences
- Speech signals
- Financial time series

Unlike feedforward networks, RNNs maintain a hidden state:

$$h_t = f(h_{t-1}, x_t), output : y_t = g(h_t)$$

- x_t : input at time t
- h_t : hidden state (memory) at time t
- h_{t-1} : previous hidden state

Key Idea: The hidden state acts as a memory that carries information from previous time steps.

Limitations of RNNs

1. Sequential Computation
Cannot parallelize across time:

$$h_t \text{ depends on } h_{t-1}$$

2. Long-Range Dependency Problem
Information from early tokens decays exponentially.

3. Optimization Difficulty
Gradient instability over long sequences.

$(w_i = w_i + \eta.(c(x) - h(x)).x_i)$ (reference)

Result: Motivated LSTM, GRU, and eventually Transformers [1].

Transformer Architecture

Introduced in: 2017.

Key idea: self-attention [2].

> *Instead of processing words sequentially, the model examines relationships between all words simultaneously.*

Advantages:

- parallel computation
- long-range context
- scalable architectures

Example: Input Sentence

Sentence: "The cat sat"

Token Embeddings:[1]

$$X = \begin{bmatrix} \text{cat} \\ \text{on} \\ \text{mat} \end{bmatrix}$$

- Each word is converted into a vector
- Forms matrix $X \in \mathbb{R}^{3 \times d}$, three words each with d-dimensions

[1] Embeddings are used to represent words as dense vectors that capture semantic relationships, enabling machines to understand and process language effectively.

Step 1: Sentence to Tokens

Sentence:

$$\text{“cat on mat”}$$

Tokens:

$$[\text{cat},\ \text{on},\ \text{mat}]$$

Number of tokens:

$$n = 3$$

Tokens $\rightarrow$ IDs (Vocabulary mapping)
cat $\rightarrow$ 101,on $\rightarrow$ 52, mat $\rightarrow$ 176

Step 2: Embedding Matrix X

Text $\rightarrow$ Tokens $\rightarrow$ IDs $\rightarrow$ Embeddings $\rightarrow X$

IDs $\rightarrow$ Embeddings

Embedding matrix: $E \in \mathbb{R}^{|V| \times d}$

$|V|$: vocabulary space, d: Embedding dimension.

Assume embedding dimension $d = 3$

$$X = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

Rows:

- cat $= [1, 0, 1]$
- on $= [0, 1, 1]$
- mat $= [1, 1, 0]$

Step 3: Weight Matrices

$$W^Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \quad W^K = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$W^V = \begin{bmatrix} 1 & 0 \\ 0 & 2 \\ 1 & 1 \end{bmatrix}$$

How they are obtained?

- At the start of training: Values are small random numbers.
- Those matrices are learnable weight matrices in the self-attention mechanism of a transformer.

Step 4: Compute $Q = XW^Q$

Query (Q): what this token is looking for

$$Q = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \\ 1 & 1 \end{bmatrix}$$

Step 5: Compute $K = XW^K$

Key (K): what this token offers / represents

$$K = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 1 & 1 \\ 2 & 1 \end{bmatrix}$$

Step 6: Compute $V = XW^V$

Value (V): the actual information content to pass along

$$V = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 2 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 3 \\ 1 & 2 \end{bmatrix}$$

Step 7: Compute QK^T

Compute Similarity:

$$K^T = \begin{bmatrix} 1 & 1 & 2 \\ 2 & 1 & 1 \end{bmatrix}$$

$$Q.K^T = \begin{bmatrix} 2 & 1 \\ 1 & 2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 2 \\ 2 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 3 & 5 \\ 5 & 3 & 4 \\ 3 & 2 & 3 \end{bmatrix}$$

Step 8: Attention Scores

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q.K^T}{\sqrt{d_k}}\right).V$$

- $Q.K^T$ gives similarity scores
- softmax converts them into probabilities
- Multiply with V to get final output
- $\sqrt{3} = 1.732.$ and V computed above.

Final Attention Weights

- Shows how much each word attends to others: Rows = current word Columns = words it attends to

$$\text{Softmax}(Q.K^T) = \begin{bmatrix} 0.245 & 0.090 & 0.665 \\ 0.665 & 0.090 & 0.245 \\ 0.422 & 0.155 & 0.422 \end{bmatrix}$$

Interpretation:

- "cat" attends most to "mat"
- "on" attends most to "cat"
- "mat" attends to both "cat" and itself
- The model builds relationships between words.

ChatGPT

How ChatGPT works

- Each word attends to all others
- Important words get higher weights
- Meanings are combined using these weights

This is how context is understood.

Self-Attention: computes how each word relates to all other words, assigns importance weights, and builds a new context-aware representation of the sentence.

Given input (sentence embeddings:a matrix representing n words, each described by a d-dimensional vector):

$$X \in \mathbb{R}^{n \times d}$$

Linear projections ($W^Q \in \mathbb{R}^{d \times d_k}$):

$$Q = X.W^Q, \quad K = X.W^K, \quad V = X.W^V$$

Self-attention computes dependencies between tokens.

$$Attention(Q, K, V) = softmax\left(\frac{Q.K^T}{\sqrt{d_k}}\right).V$$

Q = queries, K = keys, V = values: Attention weights determine how strongly words interact.

Step 1: Initialization

- W^Q, W^K, W^V are initialized randomly
- They Learned from data via training
- They encode:
  - What to attend to (Q)
  - How to match (K)
  - What information to pass (V)
- Typically small values (e.g., Gaussian or Xavier initialization)
- Example (toy case):

$$W^Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$$

- Real models use floating-point values, not simple integers

Step 2: Forward Pass

- Input embeddings: X
- Compute:

$$Q = X.W^Q, \quad K = X.W^K, \quad V = X.W^V$$

- Attention scores:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q.K^T}{\sqrt{d_k}}\right).V$$

Step 3: Loss Computation

- Model produces output (e.g., next word prediction)
- Compare with ground truth
- Compute loss:
  - Cross-entropy (common choice)
- Loss measures prediction error

Step 4: Backpropagation

- Compute gradients of loss w.r.t. weights:

$$\frac{\partial L}{\partial W^Q}, \quad \frac{\partial L}{\partial W^K}, \quad \frac{\partial L}{\partial W^V}$$

- Gradients flow through attention mechanism
- Chain rule applied through all operations

(Like a student:Makes mistake, Teacher gives feedback, Adjusts understanding)

Step 5: Weight Update

- Update weights using gradient descent:

$$W^Q \leftarrow W^Q - \eta \frac{\partial L}{\partial W^Q}$$

$$W^K \leftarrow W^K - \eta \frac{\partial L}{\partial W^K}$$

$$W^V \leftarrow W^V - \eta \frac{\partial L}{\partial W^V}$$

- $\eta =$ learning rate (reference)
- Process repeated over many training steps

Multi-Head Attention

Multiple attention heads are used.

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

$$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_h)W^O$$

Each head captures different relationships between words.
Benefits:

- Captures multiple representation subspaces
- Improves expressiveness

Positional Encoding

Unlike RNN, Transformers process tokens in parallel, so positional information must be added.

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d}}\right)$$

$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d}}\right)$$

This enables the model to capture word order.

Transformer Block

Each block consists of:

- Multi-Head Attention
- Add & LayerNorm
- Feedforward Network
- Add & LayerNorm

Feedforward:(A simple neural network applied to each position)

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Large Language Models

LLMs are transformer-based models trained on massive text datasets.
Characteristics:

- billions of parameters
- large-scale training data
- capable of many NLP tasks

BERT: Bidirectional Encoder Representations

Training Objective: Masked Language Modeling (MLM) [3]
Random masking:

$$p(x_i|x_{\setminus i})$$

Loss:

$$\mathcal{L}_{MLM} = -\sum \log p(x_i|x_{\setminus i})$$

Applications:

- Classification
- Question Answering
- NER

GPT: Generative Pre-trained Transformer [4, 5]

Autoregressive Objective:

$$p(x) = \prod_{t=1}^{T} p(x_t | x_{<t})$$

Causal Masking ensures:

$$x_t \text{ cannot attend to } x_{>t}$$

Strong for:

- Text generation
- Dialogue systems
- Code generation

Examples of large language models:

Model	Architecture	Primary Purpose	Example Tasks
GPT	Transformer Decoder	Text generation	Chatbots, text completion, translation
BERT	Transformer Encoder	Language understanding	Text classification, NER, question answering
PaLM	Large Transformer (Pathways system)	Multi-task reasoning	QA, coding assistance, reasoning tasks
LLaMA	Efficient Transformer architecture	Open research and deployment	Text generation, summarization, dialogue

These models power modern AI systems such as chatbots and code assistants.

BERT vs GPT (Encoder vs Decoder Architectures)

Encoder-only → Bidirectional context (BERT) [3]
Decoder-only → Autoregressive (GPT)
Encoder-Decoder → Seq2Seq (T5) [6]

Model	Architecture	Application
BERT	Encoder	language understanding
GPT	Decoder	text generation

Examples:

- BERT → search and QA
- GPT → conversational AI

Emergent Abilities

Large language models exhibit abilities not explicitly programmed.
Examples:

- translation
- summarization
- code generation
- reasoning tasks

Applications in NLP

- Machine Translation
- Summarization
- Dialogue Systems
- Question Answering
- Code Generation

Applications in Speech

Text $\leftrightarrow$ Speech modeling
Speech Recognition: $p(\text{text}|\text{audio})$

Speech Synthesis: $p(\text{audio}|\text{text})$

Transformers used in:

- ASR
- TTS

Applications in Computer Vision

Vision Transformers (ViT) [7]

An image is split into patches, each Image patches as tokens, like in NLP:

$$x = (x_1, x_2, ..., x_n)$$

Pipeline:

- Image $\rightarrow$ split into patches
- Patches $\rightarrow$ embedding vectors
- Add positional encoding
- Pass through transformer encoder
- Classification head: a linear layer + softmax that converts features into final class prediction

Applications:

- Image classification
- Object detection
- Image segmentation
- Image generation

Multimodal Models

Joint modeling: learning relationships between text and image together

$$p(x_{text}, x_{image})$$

Applications:

- Image captioning
- Visual Question Answering
- Text-to-image generation

Limitations of LLMs

- **Hallucination**: It is generation of false or unsupported outputs by a model while presenting them as factual.
- **Bias amplification**
- **Lack of reasoning consistency**
- **High computational cost**
- **Data privacy concerns**

Prompt: Who discovered gravity in 1905?

Hallucinated answer:Albert Einstein discovered gravity in 1905.

Correct fact: Gravity theory was formulated by Isaac Newton and later expanded by Albert Einstein through General Relativity in 1915.

Mathematical Limitations of LLMs

- **No explicit symbolic reasoning**
- **Approximate next-token predictor**

Model only optimizes: $\arg\max p(x_t|x_{<t})$

Not necessarily: Truth or factual correctness

Ethical Issues

Important concerns:

- fairness and bias
- misinformation
- privacy
- responsible AI development

Ensure AI is fair, safe, accurate, and respects user privacy

Future Research Directions

- Efficient transformers
- Retrieval-augmented generation
- Neuro-symbolic models
- Trustworthy and explainable AI
- Smaller domain-specific LLMs

Future work focuses on making AI efficient, reliable, explainable, and more specialized

Key Points

- Generative AI is transforming computing.
- Large language models are based on transformers.
- Scaling data and models improves performance.
- Responsible development is crucial.

Conclusion

- Transformers revolutionized AI
- Scaling enabled LLM breakthroughs
- Applications span NLP, Speech, Vision
- Significant open research challenges remain

Thank You

Questions?

References I

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *NAACL*, 2019.

[4] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[5] T. B. Brown, B. Mann, N. Ryder, *et al.*, "Language models are few-shot learners," *NeurIPS*, 2020.

References II

[6] C. Raffel, N. Shazeer, *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, 2020.

[7] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *ICLR*, 2021.