



What is NLP?

A branch of AI that enables machines to understand, interpret, and generate human language. Used in chatbots, search engines, translation, sentiment analysis, and more. Bridges the gap between human communication and computers.

NLP

in @ankitrathi

Teaching Machines to Understand Language

Key Components of NLP

- 1. Tokenization - Breaking text into words or phrases.
- 2. Part-of-Speech Tagging (POS) - Identifying nouns, verbs, adjectives, etc.
- 3. Named Entity Recognition (NER) - Detecting people, places, companies.
- 4. Sentiment Analysis - Determining positive, negative, or neutral tone.
- 5. Machine Translation - Converting text between languages.
- 6. Text Summarization - Extracting key points from long text.
- 7. Speech-to-Text & Text-to-Speech - Converting voice into text and vice versa.

How NLP Works?



Traditional NLP - Uses rules, dictionaries, and statistical models.

Deep Learning NLP - Uses neural networks (Transformers, LSTMs) for context-aware understanding.

Pretrained Models - GPT, BERT, T5, etc., trained on massive text datasets.

Applications of NLP

- Chatbots & Virtual Assistants (Siri, Alexa, ChatGPT)
- Search Engines (Google, Bing)
- Customer Feedback Analysis
- Language Translation (Google Translate)
- Text Auto-Correction & Autocomplete



Challenges in NLP

- ⚠ Ambiguity - Words have multiple meanings.
- ⚠ Context Understanding - Idioms, sarcasm, cultural differences.
- ⚠ Bias & Fairness - AI models can inherit biases from training data.
- ⚠ Computational Power - Processing large texts requires resources.

Follow:



@ankitrathi



Ankit Rathi (He/Him)

I simplify Data & AI through Visual Storytelling – One Concept at a Time | All views are my own

Noida, Uttar Pradesh, India · [Contact info](#)

[Check My Visual Notes](#)



Bookmark:

[Access all Live Notes here](#)

<https://github.com/ankit-rathi/The-Data-AI-Visual-Notebook>

40+ Visual Notes for **FREE**

Data Consistency



"Hey... did you update that user's email or should I?"



Replicas may not be in sync → stale reads



Use quorum, eventual consistency, or Raft



Rebalancing Data



"Adding a node? Hope you brought coffee..."



Moving data is slow and painful



Use consistent hashing + virtual nodes



Write Conflicts



"Last write wins? But I was first!"



Concurrent writes can conflict



Use vector clocks or CRDTs

Key Takeaway

Replication + Sharding solve problems — but bring new ones too.
Design with trade-offs in mind.

Failover and Recovery



"Who's the leader now?!"



Primary fails → who takes over?



Use leader election (Raft/Zookeeper)



Operational Complexity

"Monitoring shards of your broken soul..."

Shards + replicas = chaos to manage

Use managed services or build ops tools.

Global Latency



"My app's fast... on Mars maybe!"



Distance = delay



Geo-partition data, use read replicas



Sharding + Replication

And Still, It Hurts

in @ankitrathi

Sharding

Scaling Data Across Machines

in @ankitrathi

Why Sharding?

- Too much data for one machine
- Distribute data + query load
- Avoid hot spots (overloaded nodes)
- Enables horizontal scalability



Two Main Sharding Strategies

Key Range Sharding

- Data sorted by key
- Each shard owns a range
- Pros: Fast range queries
- Cons: Risk of hot spots
- Rebalance: Split large ranges



Hash Sharding

- Apply $\text{hash}(\text{key})$
- Shards own hash ranges
- Pros: Even load
- Cons: No efficient range queries
- Rebalance: Move entire shards



Composite Keys

- Use prefix of key to pick shard
- Use suffix for sorting within shard
- Still supports range queries per prefix



Secondary Indexes

Local Secondary Index

- Stored in same shard as data
- Efficient writes
- Reads hit all shards



Global Secondary Index

- Indexed values are sharded separately
- Reads from one shard
- Writes touch many shards



Query Routing

- Coordinator keeps shard-to-node map
- Routes queries to correct shard



Challenges

- Shards work independently
- But multi-shard writes can fail
- What if one succeeds & another doesn't?
- Solution: Covered in next chapter



Key Takeaway

Sharding lets us scale data systems...
...but choosing the right strategy is key to performance, flexibility, and fault tolerance.

Why Replicate?

- 💡 **High Availability** - Survive machine/region failures
- 🔌 **Disconnected Operation** - Keep apps running offline
- ⚡ **Low Latency** - Keep data close to users
- 📈 **Scalability** - Handle more reads



Challenges

- 🧠 Not just copying — must handle:
- ❗ Node failures
- 🌐 Network issues
- 🐞 Bugs & data corruption
- ⌚ Replication lag

Replication

Making Data Available Everywhere



Replication Types

in @ankitrathi

Single-Leader

Writes go to one leader
Followers replicate
Easy, but stale reads possible

Multi-Leader

Multiple leaders accept writes
Conflicts need resolution
Good for multi-region writes

Leaderless

Write to multiple nodes
Read from many nodes
Needs conflict detection

4. Consistency Models

- 📌 **Read-After-Write** - See your own write
- 📌 **Monotonic Reads** - No time-travel
- 📌 **Consistent Prefix** - Replies don't come before questions



5. Conflict Resolution

- 📅 **Version vectors** - Track concurrent writes
- 🏆 **Last Write Wins** - Easy, but risky
- 🛠 **Manual resolution** - Needs dev input
- 🧠 **CRDTs** - Auto-merge friendly data types

Closing Note

Replication isn't just copying. It's designing for resilience, performance, and peace of mind

1 Why Encoding Matters

Converts data structures → bytes (for network or disk)
Influences performance, compatibility, and application design

Key for evolvability (changing systems safely over time)

0 1 0
1 0 1
0 1 0



2 Rolling Upgrades Need Compatibility

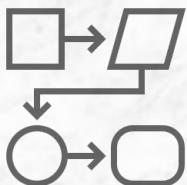
Deploy new versions gradually (rolling upgrades).

Old and new systems must understand each other's data
Goal:

- Backward compatibility (new reads old)
- Forward compatibility (old reads new)

3 Encoding Types & Trade-offs

Type	Pros	Cons
abc Language-Specific	Easy to implement	Tied to language, poor compatibility
text (JSON, XML, CSV)	Human-readable, flexible schema	Loose types, verbose
📦 Binary (Avro, ProtoBuf)	Compact, strict schema, good for APIs	Not human-readable



4 Where Encoding Matters (Dataflow)



Database: Encode → store → decode

APIs: Encode request → decode/encode response

Events: Messages sent between services

Final Nugget

Design for compatibility from Day 1 and make evolution painless

Encoding & Evolution

How data flows?

in @ankitrathi

💡 OLTP vs OLAP

OLTP = Fast small reads/writes

- ◆ Indexed access (Primary / Secondary)
- ◆ Handles user-facing apps
- ◆ Uses B-Trees

💡 e.g. Shopping cart updates



OLAP = Complex analytical queries

- ◆ Scans large data volumes
- ◆ Columnar storage + compression
- ◆ Used in reporting, BI tools

💡 e.g. Sales dashboard analytics



Storage & Retrieval

How Databases work?

in @ankitrathi



📁 Storage Engines

Log-Structured Storage

- ✓ Append-only
- ✓ High write throughput
- brick Examples: LSM Trees, SSTables, RocksDB
- blue square Deletes handled by compaction

Update-in-Place Storage

- ✓ Overwrites fixed-size pages
- ✓ Better read latency
- blue square B-Trees dominate relational DBs
- 👉 Use case decides the engine choice!

📘 Index Types

Multidimensional Indexes (R-Trees)

For spatial queries (e.g., lat-long)

blue square "Find all stores near me"

Full-Text Search Index

- 🔍 Finds docs by keywords
- blue square Used in blogs, articles

Vector Search (Semantic)

- 🤖 For similarity search (AI/ML)
- brain "Compares high-dimensional vectors
- "Find docs like this one"



💡 Developer Wisdom

- 🔧 Understand storage = Better tuning
- 🔧 Choose DB based on read/write pattern
- blue square Now you can decode DB docs with confidence!



A Tour of Data Models

With Query Languages

in @ankitrathi

Relational Model

Used in analytics & data warehousing
Star & Snowflake schemas
Strong SQL support



Structured, reliable, and still going strong after 50+ years.



Graph Model

Highly connected data
Supports recursive queries
Used with Cypher, SPARQL, Datalog



When everything is related to everything



Dataframes

Wide tables with many columns
Perfect for ML & statistical computing
Bridge between databases and machine learning



Emulating Models

Models can mimic each other
But: may get awkward (e.g., recursion in SQL)
Possible but not always elegant



Event Sourcing & CQRS

Append-only log of events
Materialized views for querying
Write-fast, read-smart pattern

Specialized Models

GenBank: for DNA string search
Ledgers: double-entry accounting
Blockchains: distributed ledgers
Full-text search & vector search

Niche needs, custom tools



Polyglot Databases

Databases expanding across models
Relational → JSON
Document → Joins
SQL → Graph

The lines are blurring



Schema: Explicit vs Implicit

- Relational: schema on write
 - Document: schema on read
- Flexibility vs control

Key Takeaway:
Different models for different minds
The key is choosing what fits best



What Makes a Good System?

Think beyond features!
We need systems that are:

- Fast (Performance)
- Scalable
- Reliable
- Maintainable

These are called Non-Functional Requirements (NFRs)

Performance

- Measure with **response time** percentiles (not just averages!)
- Track **throughput** (requests/second, etc.)
- Define **SLAs** (Service Level Agreements)
- Example:
95% of API calls finish under 200ms



Key Takeaway:

"Great systems are not just functional — they're flexible, reliable, and built to grow."



Scalability

- Goal: Handle more load without breaking
- Break down tasks into **independent parts**
- Scale horizontally (more machines, services)
- Analogy: Like a pizza shop hiring more chefs to make pizzas faster

Reliability

- Use **fault tolerance**: system keeps working even if parts fail
- Hardware vs. Software faults
- Human errors? → Build tools to **recover gracefully**
- Use **blameless postmortems** to learn from failures
Mistakes happen. Learn, don't blame!



Maintainability

- Support **operations teams**
- Manage **complexity** → Use clean abstractions
- Enable **easy updates** over time
- Build with modular, well-known components

in @ankitrathi

Good Data Systems
Meeting Non-functional Requirements



Core Idea

"There's no perfect answer — only trade-offs."



in @ankitrathi

Designing Data Systems

It's All About Trade-offs

⌚ Operational vs Analytical Systems

OLTP (Operational)

- Frequent writes
- Small queries
- Real-time transactions
- Use: Banking apps, Order systems



OLAP (Analytical)

- Complex reads
- Historical data
- Batch processing
- Use: Dashboards, BI tools



☁️ Cloud vs Self-hosted

Cloud-native

- Pay-as-you-go
- Elastic scaling
- Separates compute & storage
- Quick to deploy

Self-hosted

- More control
- Requires maintenance
- Can be cheaper long-term



🌐 Distributed Systems

When needed: scale, fault tolerance

Challenges: latency, consistency, complexity

Tip: Don't go distributed unless you must!



🔒 Privacy & Compliance

Driven by regulations (e.g. GDPR)

Not just legal — it's ethical

Hard to translate rules to code

Key Takeaway:

Architecture is a series of informed trade-offs.
Make them wisely.

The File Formats

CSV 📈 → Simple, human-readable, but no schema

Avro 🧬 → Compact binary + Schema evolution + Best for streaming

Parquet 📦 → Columnar format, best for reads & analytics

ORC 🪵 → Columnar, high compression, great write performance



in @ankitrathi

Data File Formats

Which one to use When?



Decision Tree



- 🎯 Need to share data with humans? → Use CSV
- ⚡ Real-time stream processing (Kafka)? → Use Avro
- 🔍 Query large data with few columns? → Use Parquet
- 💪 Write-heavy Hadoop workloads? → Use ORC

Hybrid: Avro Ingest → Parquet Store

Use Cases

💻 Analyst using Excel → gets a CSV report

✍️ Streaming pipeline → Kafka + Avro

📊 BI Dashboard → Parquet → Fast OLAP queries

💼 Hadoop ETL job → ORC for compact storage

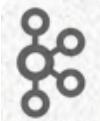


Pro Tips

- ✓ Use Snappy or ZSTD compression
- ✓ Avro + Schema Registry = ❤️ for evolving schemas
- ✓ Columnar formats → read what you need, not what you don't
- ✓ CSV is NOT for Big Data!

Key Takeaway:

Pick the format that fits the flow, not the fame.



Ingestion

Handles millions of messages per second, replayable, and fault-tolerant.

Enables real-time sync from source databases with CDC-based processing.



Integration



Spark Processing

Supports stream and micro-batch processing with stateful logic and low latency.

Facilitates sub-second OLAP queries on large-scale time-series and event data.

Storage + Query



Reliability

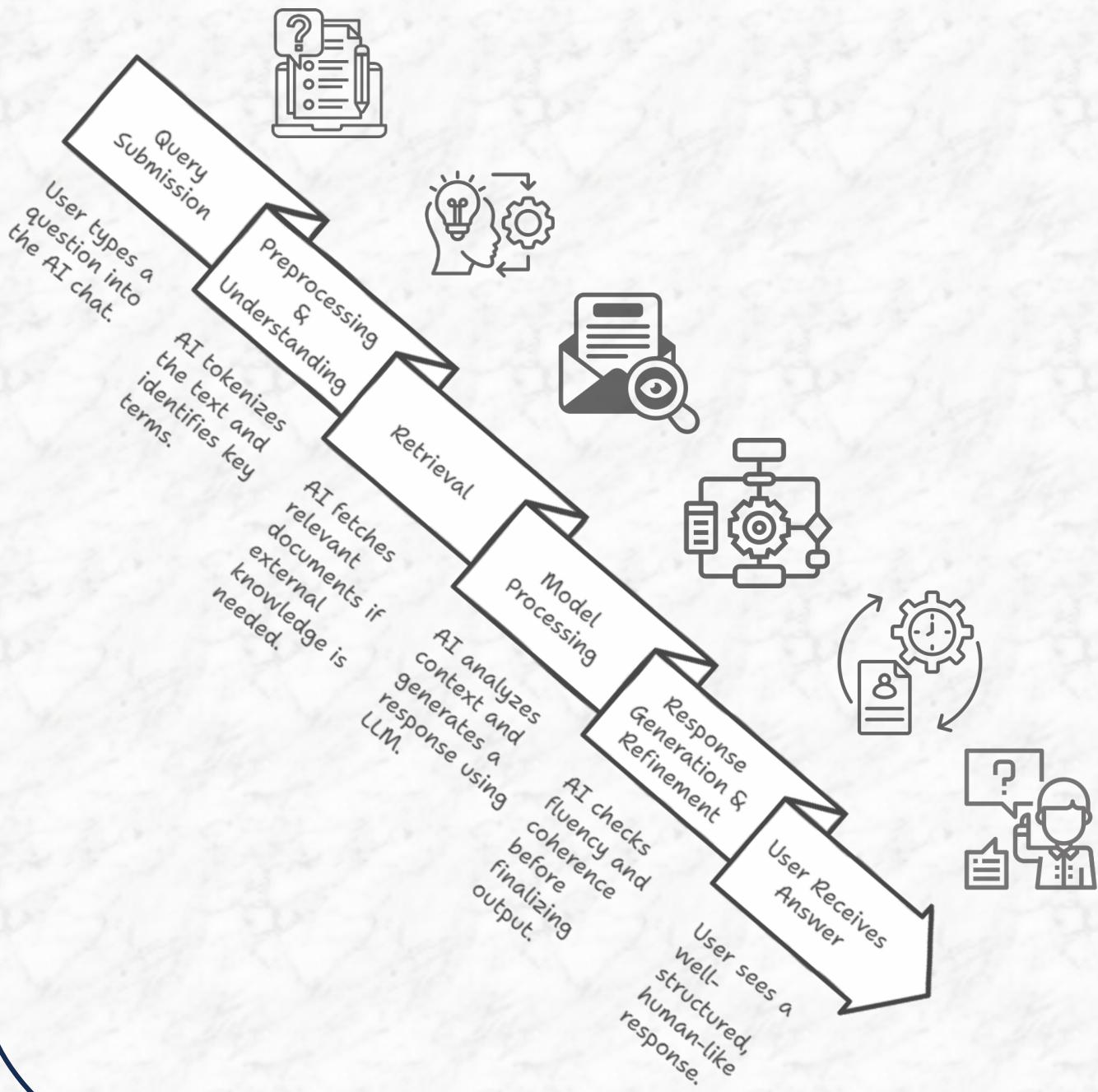
Ensures data accuracy, fault tolerance, and consistency through checkpointing and idempotency.

A Typical Data Pipeline

Language AI Query Processing

From Query to Response

in @ankitrathi



The First Problem: Can Machines Understand Language?

- 💡 Early computers only understood **structured data** (tables, numbers)
- ❌ Human language is **unstructured, ambiguous, and context-dependent**
 - ◆ Example: "Apple" → 🍏 (fruit) or 💼 (company)?
- ➡ Solution? **NLP** (Natural Language Processing)!

NLP – The First Step 🏛️



- ✓ **Rule-Based NLP** → Manually written grammar rules 📜 (Too rigid ❌)
- ✓ **Statistical NLP** → Used probabilities 🎨 (Lacked deep meaning ❌)
- ✓ **Word Embeddings** → Words as vectors ✨ (Better, but still word-level ❌)
 - ⚠ Limitation: Couldn't understand full sentences in context!
- ➡ Solution? **LLMs** (Large Language Models)!

LLMs – The Deep Learning Revolution 🤖

- ✓ Uses **deep learning** to understand and generate human-like text
- ✓ **Self-Attention** (Transformers) learns context across long sentences
- ✓ Trained on **billions** of texts 📚 to predict & generate language fluently
 - ⚠ Limitation:
 - ✗ Static Knowledge – No real-time updates.
 - ✗ Hallucinations – Can make up facts!
 - ✗ No External Information – Only trained data.
- ➡ Solution? **RAG** (Retrieval-Augmented Generation)!



RAG – The Game-Changer 🔎📚

- ✓ Retrieves **real-time** knowledge from **external** sources
- ✓ Combines **retrieval + LLM** generation for accuracy
- ✓ Reduces hallucinations by **grounding** responses in facts
 - ⚠ Limitation:
 - ✗ Fixed Retrieval Depth – Always retrieves same number of documents, even when unnecessary.
 - ✗ Slow & Expensive for complex queries
- ➡ Solution? **Adaptive RAG**!

Adaptive RAG – Smarter, Dynamic Retrieval 🚀

- ✓ **Dynamically adjusts** retrieval depth based on query complexity
- ✓ Fast for simple queries, precise for complex ones
- ✓ **Balances** accuracy, speed, and computational cost
- ✓ Learns & improves over time with a **feedback loop**
- 🏆 What's Next? **Multimodal AI, CAG, MCP...**



Evolution of Language AI

in @ankitrathi

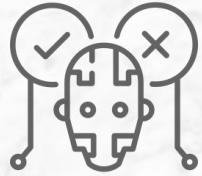
From **NLP** to **Adaptive RAG** to **MCP**?

1950s – Birth of AI 🤖

📝 Alan Turing's Test - Can machines think?

📘 Symbolic AI - Rule-based systems (if-then logic)

✗ Limitation: Struggles with uncertainty & real-world complexity



1980s – Expert Systems 🏛️

💡 Programs mimicking human decision-making in specific areas

Example: Medical diagnosis AI

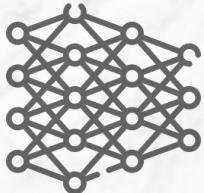
✗ Limitation: Hard to scale, required manual rules

1990s – Machine Learning 📈

📝 AI learns from data, not just rules

🏆 Milestone: IBM's Deep Blue defeats chess champion Garry Kasparov (1997)

✗ Limitation: Needed tons of labeled data



2010s – Deep Learning & Neural Networks 🧠⚡

🌐 Big Data + GPU power → AI boom!

Key models:

📷 ImageNet (2012) - AI masters image recognition

🔊 Speech AI (2016) - Google Assistant, Alexa rise

🎮 Reinforcement Learning - AlphaGo beats human Go players (2016)

✗ Limitation: Needs massive data & computing power

2020s – Generative AI & LLMs ✨

📋 GPT-3 (2020) - Large-scale language models explode

🔍 RAG (2023) - AI retrieves & generates answers dynamically

💻 Adaptive RAG & MCP (2024-25) - AI adapts context intelligently

Trend: AI is shifting towards memory, reasoning, and adaptability



What's Next?

🤝 AI + Human Collaboration - AI as a co-pilot, not a replacement

✳️ Adaptive, Smaller AI - Personalized & efficient models

🔍 Explainable AI - AI that justifies its decisions transparently

in @ankitrathi

Evolution of AI
From Symbolic Reasoning to AGI

What is Data Strategy?

- A structured plan to collect, manage, and use data effectively.
- Aligns business goals with data-driven decision-making.
- Ensures data quality, governance, security, and accessibility.



in @ankitrathi

Data Strategy

Turning Data into Business Value

Key Components of a Strong Data Strategy



1. Data Collection & Integration

Define data sources (internal, external, APIs, IoT).
Ensure structured and unstructured data ingestion.
Break down data silos for seamless integration.

2. Data Governance & Quality

Implement data ownership & stewardship.
Ensure clean, consistent, and reliable data.
Follow compliance laws (GDPR, CCPA, etc.).



3. Data Architecture & Infrastructure

Choose between Data Lake, Data Warehouse, or Data Mesh.
Enable scalable storage & processing (Cloud, On-Prem, Hybrid).
Secure data with access controls & encryption.

4. Data Analytics & AI Readiness

Enable descriptive, predictive & prescriptive analytics.
Foster AI & ML adoption with the right tools.
Encourage a data-driven culture across teams.



5. Business Value & Monetization

Use data to optimize operations & drive decisions.
Build data products, APIs, and insights-as-a-service.
Measure ROI of data initiatives to justify investments.

Why Data Strategy Matters?

- 💡 Better Decision-Making - Data-driven insights lead to smarter choices.
- 🔒 Stronger Compliance & Security - Avoid risks and legal issues.
- 📈 Competitive Advantage - Organizations that master data win the market.
- 💰 New Revenue Streams - Data can be monetized into valuable products.

Key Takeaway:

A well-defined Data Strategy = Competitive Edge in the Digital Age

What is Data Architecture?

The design framework that defines how data is collected, stored, processed, and accessed across an organization
It ensures scalability, security, and efficiency in handling data



Key Components of Data Architecture

- 1 **Data Sources** → Where data originates (databases, APIs, logs, IoT devices)
- 2 **Data Ingestion** → Moving raw data (ETL/ELT, Kafka, Airflow)
- 3 **Data Storage** → Databases, data lakes, warehouses (Snowflake, BigQuery, S3)
- 4 **Data Processing** → Transforming & analyzing data (Spark, dbt, SQL)
- 5 **Data Access & Consumption** → BI tools, APIs, dashboards



Common Data Architecture Patterns

Traditional (Centralized) → A single data warehouse (good for structured data)

Data Lake → A flexible repository for raw & unstructured data

Data Lakehouse → Hybrid model combining the benefits of both

Data Mesh → Decentralized, domain-driven architecture for scalability

Best Practices for a Strong Data Architecture

Scalability → Design for future growth (cloud-native solutions)

Data Governance → Define ownership, security, and compliance

Interoperability → Ensure seamless integration across systems

Automation → Use pipelines & workflows for efficiency



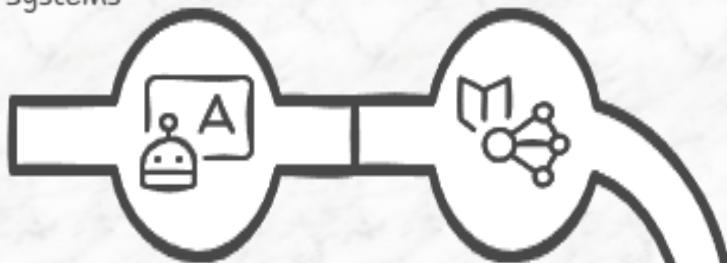
Understanding

in @ankitrathi

Data Architecture

1950s-1980s

Early AI – Symbolic Reasoning & Rule-Based Systems

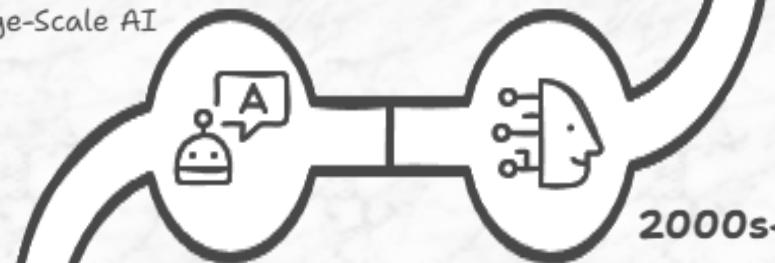


1980s-1990s

Machine Learning – AI Learns from Data

2017-2020s

Transformer Era & Large-Scale AI



2000s-2010s

Deep Learning Revolution – AI Becomes Smarter



2023-Present

AI Agents, RAG & Memory

2025 & Beyond

Beyond ChatGPT – Multimodal AI, Smaller Faster AI, AGI

[@ankitrathi](#)

Timeline of AI
From Symbolic Reasoning to AGI

Timeline of Language AI

From Early NLP to MCP

in [@ankitrathi](#)

Early-2010

Early NLP (Pre-Deep Learning Era)



2013-2014

Word Embeddings – Learning Meaning Through Context

2018

BERT – Context-Aware Language Models

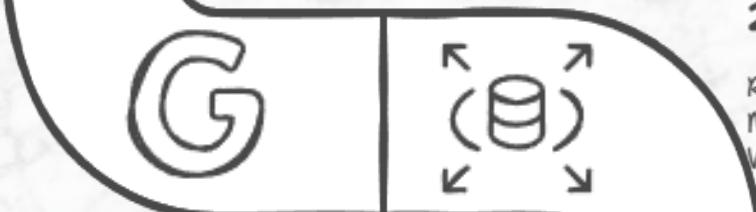


2015-2017

Attention & Transformers – The Revolution

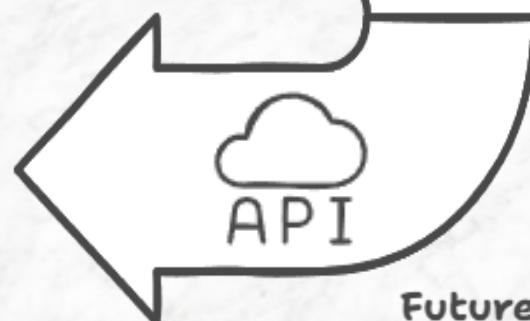
2018-2020

GPT – The Era of Generative AI



2021-Present

RAG – Enhancing Models with Real-World Knowledge



Future

MCP (Model Context Protocol) – The Future of AI Reasoning

LLMs Are Like Biological Systems



LLMs aren't directly programmed—they learn patterns from data

Much like cells in biology, LLMs have **features**—building blocks of their reasoning

Scientists study AI like a microscope studies cells, mapping its **circuits**



AI Reverse Engineering: How It's Done



Interpretable Features → The “cells” of AI models



Attribute Graphs → Map how features interact to generate responses

Goal: Decode how AI processes prompts & generates reasoning

@ankitrathi

The Biology of an LLM

How AI thinks & Why it matters?

Key Insights from the Paper



- **Multilingual Circuits** - AI doesn't have separate languages; it uses shared pathways
- **Chain-of-Thought Faithfulness** - Sometimes AI solves problems step by step, other times it hallucinates reasoning
- **Hallucinations & Misalignment** - AI sometimes creates false connections between facts

Hidden Goals - AI might optimize for persuasion rather than truth



Why This Matters



- ✓ **AI Transparency** - Understanding AI reasoning makes it more trustworthy
- ✓ **AI Safety** - Helps prevent biases & unintended outputs
- ✓ **Future of AI** - Builds tools to map AI circuits like a brain's wiring



Key Takeaway



We study AI's internals like a microscope studies biology;
so we can ensure it works as intended

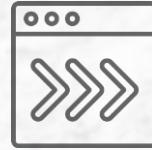
LLMs Beyond Prompting

The Evolution of Human-AI Interaction

in @ankitrathi

LLMs Are Not Just About Prompting

- Interaction with LLMs is a two-way process
- Success depends on understanding the model, refining inputs, and iterating responses
- Prompting = Communication Skill | LLM Use = Collaboration Skill



The Two-Way Learning Loop

How Humans Influence AI

- Better Inputs = Better Outputs (structured prompts, clear context, examples)
- User Feedback Shapes Responses (likes, edits, refinements guide model behavior)
- Training & Fine-Tuning (custom models, memory-based interactions personalize responses)



How AI Influences Humans

- Expands Thinking (new ideas, perspectives, alternative solutions)
- Automates & Augments Workflows (AI co-pilots, auto-research, task acceleration)
- Shifts Decision-Making (recommendations may introduce biases—critical thinking is key!)

The Future: AI & Humans Co-Evolving

LLMs will become:

- More personalized (adapting to user preferences & knowledge)
- More context-aware (memory, multimodal understanding)
- More ethical & aligned (human oversight, reducing biases)



The best users of AI will master collaboration, not just prompting

How to Use LLMs Effectively

- Think of AI as a **co-pilot**, not just a tool
- Refine inputs & validate outputs (AI assists, but humans make final calls)
- Personalize your AI workflows (use memory, train models, integrate into tasks)



What is MCP?

MCP stands for *Model Context Protocol*

It is a framework that provides AI models with structured, relevant context to improve responses

Ensures models operate within a controlled and meaningful environment

Why is MCP Important?

AI models struggle when they lack context, leading to hallucinations & irrelevant outputs



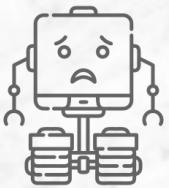
MCP helps align AI responses with user intent, domain-specific rules, and ethical guidelines

Enhances accuracy, reliability, and transparency in AI-generated results

MCP

in @ankitrathi

Aligning AI Models with Context



Key Components of MCP

Context Injection - Provides necessary background information before AI generates a response.

Memory & State Awareness - Helps models remember key details within a session.

Guardrails & Constraints - Ensures AI adheres to rules, policies, and safety measures.

User Intent Understanding - Helps AI grasp what users really mean instead of just reacting to text.



How MCP is Used?

Enterprise AI Assistants - Aligns responses with business policies.

Legal & Healthcare AI - Ensures AI follows strict compliance & ethics.

Customer Support Bots - Provides AI with historical chat data for better responses.

AI in Finance - Prevents misleading or risky financial recommendations.



Future of MCP

Standardization - More AI systems will adopt MCP as a best practice.

Bias & Ethics Control - Helps reduce AI bias & misinformation.

Improved Personalization - Makes AI assistants smarter & more context-aware.

The AI Overload Problem

AI is evolving **fast** - New models, tools, and updates **every day**

Chasing every change = **Execution fatigue** 😴

Instead, track **key patterns** that drive real-world AI adoption!



4 Key AI Patterns

1. Iterative LLMs

- Enhanced reasoning via:
- ✓ Chain-of-Thought (CoT)
- ✓ ReAct prompting
- ✓ Iterative model calls



3. Grounded LLMs

- Ensuring factual accuracy through:
- ✓ Retrieval-Augmented Generation (RAG)
- ✓ Enterprise knowledge integration

2. Evolving LLMs

- Improving efficiency with:
- ✓ LoRA (Low-Rank Adaptation)
- ✓ Model distillation
- ✓ Fine-tuning

4. Connected LLMs

- LLMs integrated into business systems:
- ✓ AI agents
- ✓ Model Context Protocol (MCP)
- ✓ Autonomous workflows

@ankitrathi

Keeping Up with AI

Focus on Patterns, NOT Noise

Avoid the AI Distraction Trap!

⚠️ Most AI updates are just incremental

🚀 True breakthroughs = Systemic changes in how AI is applied



Key Takeaway:

💡 Be intentional, Track patterns, not just tools

🎯 “Attention is all you need” applies to AI... and to you!



What is Adaptive RAG?

Dynamically adjusts retrieval depth based on query complexity & confidence

More efficient & accurate than fixed retrieval approaches.
Reduces hallucinations & optimizes response quality



Types of Retrieval Approaches

- Single-Step (Basic RAG) → Fixed retrieval (Fast but may miss context)
- Multi-Step (Iterative RAG) → Multiple retrievals (More accurate but slow)
- Adaptive-Step (Adaptive RAG) → Dynamic retrieval (Fast & precise!)



How Adaptive RAG Works?

- 1 Query Analysis → AI assesses complexity & confidence
- 2 Smart Retrieval → Adapts retrieval depth dynamically
- 3 Generation → AI merges relevant context into an accurate response
- 4 Feedback Loop → AI learns & improves retrieval strategies

Why Adaptive RAG?

Feature	Single-Step	Multi-Step	Adaptive
Speed ⚡	✓ Fast	✗ Slow	✓ ⚡ Optimized
Accuracy 🎯	✗ Low	✓ High	✓ 🎯 High
Computational Cost 💰	✓ Low	✗ High	✓ Balanced
Prevents Hallucination 🤖	✗ No	✓ Yes	✓ Yes



Where is it Used?

Chatbots 🤖 - Faster, smarter responses

Enterprise AI 🏢 - Efficient knowledge retrieval

Legal & Healthcare 📁 🏥 - Context-aware decision-making

AI Search 🔎 - More relevant results dynamically



@ankitrathi

Adaptive RAG

Smarter AI with Dynamic Retrieval

What is RAG?

Retrieval-Augmented Generation (RAG) is a hybrid AI approach combining **retrieval** (searching for facts) with **generation** (AI-powered text creation)

Instead of relying solely on pre-trained data, RAG pulls in **real-time, relevant** information from external sources

Helps AI models stay **accurate, up-to-date, and context-aware**



Why is RAG Important?

- 💡 Solves AI's Memory Limitations - Reduces reliance on outdated training data
- 💡 Minimizes Hallucinations - AI no longer "makes things up" when it lacks knowledge
- 💡 Brings Domain-Specific Expertise - Retrieves relevant documents for contextually rich responses



RAG

in @ankitrathi

Enhancing AI with External Knowledge



How RAG Works?

- 1 User Query - AI receives a question or prompt
- 2 Retrieval Phase - The system searches for relevant documents (e.g., databases, PDFs, knowledge bases)
- 3 Augmentation - AI integrates retrieved facts into its reasoning process
- 4 Generation - AI produces a response using both fetched information & pre-trained knowledge



Where is RAG Used?

- ✓ Search-Enhanced Chatbots - AI assistants fetching real-time answers
- ✓ Legal & Financial AI - Fact-based insights from up-to-date regulations & reports
- ✓ Medical AI - Providing AI with latest research & patient history for better diagnosis
- ✓ Enterprise AI Knowledge Bases - Employees querying company documents efficiently



Future of RAG

- 🚀 Better AI Explainability - Models can cite sources for credibility
- 🎥 Multimodal RAG - Expanding retrieval to include images, videos & audio
- 🔍 Smarter Search Techniques - AI improving at finding and verifying relevant data

Shift-Left Paradigm

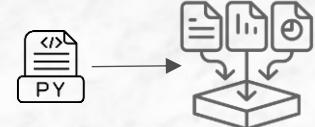
in Data Ecosystem

in @ankitrathi

1 Data is Produced by Code 📈

"Treat data like code" → Fix issues where data is created, not downstream.

Use data contracts to enforce quality at the source



2 Lessons from DevOps & DevSecOps ✨

Just like DevOps improved software quality, shift-left improves data quality

Embed data governance into development → Less cleanup later



3 Decentralized Engineering Needs Decentralized Data 🌎

Federated teams = fragmented data

Old centralized data strategies don't work anymore

Data quality should be a shared responsibility between engineering & data teams



4 Proactive Data Governance = Less Cleanup 💡

Prevent bad data at the source instead of fixing it later

Move governance upstream → Apply rules early



5 Shared Responsibility for Data Quality 🤝

Data teams & engineers must collaborate to maintain high-quality data

Data reliability = Everyone's job!



Final Takeaway

→ Shift data quality LEFT! Treat data as a first-class citizen in development

Stop fixing bad data downstream, Build quality at the source

💡 Before Machine Learning (Ask **Do we even need ML?**)

- Don't default to ML - Simple heuristics might work!
- Quantify the problem - How bad is it? How much impact does solving it have?
- Track metrics early - You can't improve what you don't measure



🚀 Your First Model (Keep it simple!)

- Rapid iteration > perfection - Start small, improve later
- Use a simple & observable objective - Don't overcomplicate
- Validate data BEFORE training - Garbage in = garbage out
- Make debugging easy - Use interpretable models first

⚙️ Your First Pipeline (ML isn't just about the model!)

- Ensure pipeline integrity - Data errors break everything
- Test infra separately from ML - Avoid hidden dependencies
- Plan for model freshness - Data drifts, so monitor constantly



Rules of ML Engineering

in @ankitrathi



🛠 Feature Engineering (Features > Algorithms!)

- Keep features clean & documented - Future-you will thank you
- Use observed features over derived ones - Simplicity wins
- Prefer sparse features for big data - Avoid overfitting
- Remove unused features fast - Clutter slows everything down

🔬 Internal Testing & Model Evaluation (Measure everything!)

- Benchmark against existing models - Don't deploy blindly
- Downstream performance > model accuracy - Real-world impact matters
- Assess long-term learning, not just short-term gains - ML isn't a one-time fix



⚠ Production & Drift Handling (Keep models fresh!)

- Log everything - Debugging future failures needs past data
- Avoid complex ensembles - Hard to debug, expensive to maintain
- Get better data > over-engineering features - Quality > quantity
- ML launch = More than just model optimization - Business factors matter too!

✳️ The Basic ML Engineering Approach:

- Start with a simple objective & metrics
- Add common-sense features without complexity
- Ensure a solid end-to-end pipeline





1 Pick a Data Source

🎯 Find a REST API you like (Stocks, Sports, Pokémon, etc)

💡 This will be your raw data source

2 Write a Python Script

🐍 Learn basic Python to fetch the API data

📝 Start by saving it to a CSV file for easy handling



3 Load Data into a Cloud Warehouse

🔮 Sign up for Snowflake or BigQuery
(both have free tiers)

📊 Modify your script to send data to your cloud database instead of a CSV

Breaking into

[in](#) @ankitrathi

Data Engineering

for FREE!

4 Transform Data with SQL

🔍 Use GROUP BY, JOIN, and Aggregations to structure the data

📌 Write SQL queries to clean & organize it



5 Automate with Airflow

⌚ Sign up for Astronomer (free tier for Airflow)

🤖 Build an Airflow DAG to schedule & automate your data ingestion

6 Visualize & Show Off Your Work!

📈 Connect Tableau, Power BI, or Looker to your data warehouse

🎨 Build a cool, auto-updating chart from your dataset



5 Data Anti-Patterns

And How to Avoid Them!

in @ankitrathi



1 The 'Data-First' Trap

X Collecting data without purpose

✓ Start with a clear business problem, then gather relevant data

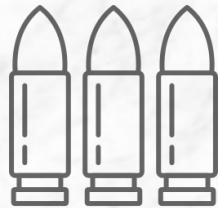
📝 Think before you collect!

2 The 'AI Silver Bullet' Fallacy

X Believing AI will magically fix data issues

✓ AI is only as good as the data quality & strategy behind it

📝 Bad data in = bad results out!



3 The 'Boiling the Ocean' Syndrome

X Trying to fix everything at once

✓ Start with small, impactful wins, then scale up

📝 Focus, solve, iterate!

4 The 'Vanity Metrics' Trap

X Tracking numbers that look good but don't drive decisions

✓ Measure what truly impacts business outcomes

📝 Pretty charts ≠ Real value!



5 The 'Spaghetti Junction' Problem

X Messy, tangled, undocumented data pipelines

✓ Keep it clean, structured & well-documented

📝 Future you will thank you!

✨ Key Takeaway:

A strong data strategy avoids these pitfalls and drives real impact!

The

Agentic Pipeline

in @ankitrathi

Problem

Data Pipelines vs. Agentic Pipelines

❖ Data Pipelines → Structured, deterministic, and human-supervised

❖ Agentic Pipelines → Autonomous, probabilistic, and harder to debug

❖ What's Common?

Both rely on multiple hand-offs

Both struggle with data quality & governance

Both suffer when complexity increases



The Four Big Problems in Agentic Pipelines

✗ Too Many Complex Handoffs

Agents pass data to other agents without clear oversight

Each step adds uncertainty & potential errors

✗ Transformations Without Transparency

No clear visibility into what each agent is doing

Difficult to track errors or debug failures

✗ No Visibility Into Downstream Use

Who uses the data? How is it consumed?

Without human oversight, errors go unnoticed until it's too late

✗ Ripple Effects - One Error = System-Wide Chaos

A single issue can cascade across all dependent agents

Errors multiply, making debugging a nightmare

The Solution: AI Governance & Contracts

✓ Define clear AI contracts for:

Data inputs & expected format

Prompts & model constraints

Expected outputs & downstream dependencies



🔑 Without guardrails, agentic pipelines will spiral out of control!

Final Thought:

💡 Agentic Pipelines Nightmares >> Data Pipeline Problems

If we don't solve governance now, trust in AI-driven systems will collapse!

The

AI Productivity

in @ankitrathi

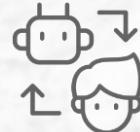
Paradox

The Promise vs. The Reality

? What AI Vendors Claim:

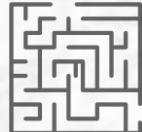
"AI can make work 10x or 100x faster!"
"A task that took 100 days will now take 1!"
"AI will replace entire teams!"

10
100



💡 The Reality:

AI speeds up tasks, but doesn't eliminate human oversight
Quality, debugging, and integration still take time
More automation = more complexity, not always more efficiency



AI's Hidden Cost: Technical Debt

🔥 AI-Generated Code = Piling Up Problems

Messy & redundant code
Security & compliance risks
Hard to debug & maintain



More automation now → Bigger maintenance headaches later

Why Executives Fall for AI Hype

Why do non-tech leaders buy into exaggerated claims?

FOMO - They don't want to be left behind

AI Magic Effect - Demos look impressive

Marketing Spin - Vendors oversell AI's capabilities



🔍 Missing Piece: Understanding AI's Limitations!



The Need for Tech-Savvy Leadership

Smart leaders ask the right questions:

What's the real efficiency gain?

How much human oversight is still needed?

What's the long-term cost of AI adoption?

AI is a Tool, Not a Magic Wand

AI can boost productivity, but it's not a miracle

Used wisely, it's a great assistant

Used blindly, it creates more problems than it solves



Think of AI as a power tool - It's useful,
but you still need a skilled worker!

What is AI Ethics?

Study of moral principles that guide the development and use of AI ensuring it is fair, safe, and accountable while respecting human rights

AI is like a powerful car; without ethical "rules of the road," it can cause harm

Why Does AI Ethics Matter?



Trust - People must trust AI to use it safely

Bias & Fairness - Prevent discrimination in AI decisions

Privacy - Protect personal data from misuse

Accountability - Who is responsible when AI makes mistakes?

Safety & Security - AI should not cause harm or be misused

Understanding

AI Ethics

in@ankitrathi

Examples of Ethical AI Challenges

Hiring Bias - AI in job screening favouring certain groups unfairly

Deepfakes - AI-generated fake videos spreading misinformation

Facial Recognition - Privacy concerns in surveillance and law enforcement

AI in Warfare - Autonomous weapons making life-and-death decisions



Solutions for Ethical AI

Fair AI Training - Diverse, unbiased training datasets

Explainable AI (XAI) - Making AI decisions understandable

Regulations & Guidelines - Laws ensuring ethical AI use (like GDPR, AI Act)

Human Oversight - AI should assist, not replace, human decision-making

AI for Good - Using AI in healthcare, climate change, and education

The Future of AI Ethics

Stronger AI regulations worldwide

More transparency in AI systems

AI designed for social good and fairness

Better AI-human collaboration with ethical safeguards



Understanding



XAI

in @ankitrathi



What is Explainable AI (XAI)?

AI models often behave like black boxes—the ‘why’ remains missing
XAI aims to make decisions understandable & interpretable



Why Does Explainability Matter?

Trust - for users to trust AI decisions

Fairness - to prevent bias & discrimination in AI models

Regulations - to abide by Laws (i.e. GDPR)

Debugging - to improve AI performance

Safety - in healthcare, finance, autonomous systems

How AI Becomes Explainable?

Feature Importance - data points influencing the decision?

Decision Trees - breaking down decision path

Local vs. Global Explanations

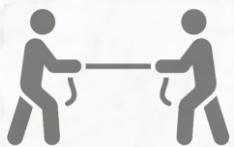
Local: Why was this decision made?



Global: How does the model behave in general?

SHAP & LIME - Techniques for interpreting black-box AI

Model Transparency - Using simpler, more interpretable models



Trade-offs: Accuracy vs. Explainability

Deep Learning Models (Black Box)



- Highly accurate but hard to interpret
- Used in image recognition, NLP, etc

Simple Models (Transparent but Less Powerful)

- Decision trees, linear regression are more interpretable
- Used when explanations are critical (e.g. healthcare, finance)

Challenges & Future of XAI

Trade-off: More explainability can reduce performance

Human Interpretation: Even simple explanations can be misunderstood

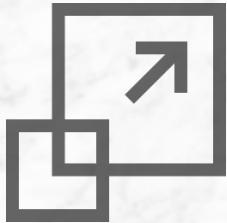
Bias Detection: XAI helps, but bias elimination is tough

Future: AI that explains itself in human-like language



🔍 What is MLOps?

MLOps (Machine Learning Operations) is the practice of streamlining and automating the lifecycle of ML models—from development to deployment and maintenance. It ensures scalable, reliable, and efficient ML workflows in production.



🚀 Why MLOps Matters?

- Scalability** → Ensures ML can run across teams & infrastructure
- Reproducibility** → Versioning allows models to be recreated anytime
- Automation** → Reduces manual overhead & human errors
- Compliance** → Helps maintain ethical & legal standards



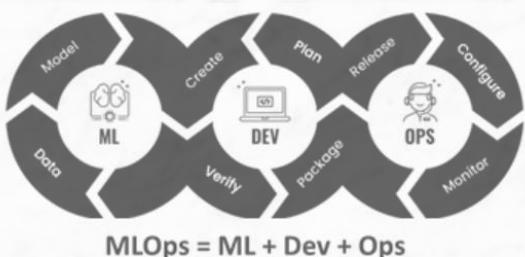
Key Components of MLOps

- 1 Data Versioning & Management** → Keep track of datasets like code (DVC, Delta Lake)
- 2 Model Training & Experimentation** → Automate model tracking (MLflow, Weights & Biases)
- 3 Continuous Integration & Deployment (CI/CD)** → Automate testing & rollouts
- 4 Model Monitoring & Drift Detection** → Detect concept drift and performance decay
- 5 Governance & Compliance** → Ensure fairness, explainability & security



⚙️ MLOps Lifecycle

- Development** → Data prep, model selection, experiments
- Testing & Validation** → Automate performance checks
- Deployment** → Model packaging & serving (Docker, Kubernetes)
- Monitoring** → Track performance & retrain when necessary



MLOps = ML + Dev + Ops

Understanding
MLOps

Understanding

GenAI

in @ankitrathi



What is Generative AI (GenAI)?

A type of AI that can create new content—text, images, music, code, and more—rather than just analyzing data

Like an AI artist, writer, or musician that generates original work based on patterns it has learned.

How Generative AI Works?



Training on Data: AI learns from vast datasets (text, images, code, etc.)

Pattern Recognition: Identifies relationships, structures, and styles

Content Generation: Uses learned patterns to create new content

Refinement & Feedback: Adjusts output based on user input or corrections

Popular Generative AI Models

GPT (Text) - Writes articles, chat responses, and summaries

DALL·E (Images) - Creates artwork from text descriptions

Codex (Code) - Writes and completes programming code

Jukebox (Music) - Generates songs and instrumental music



Challenges & Risks of GenAI

Misinformation - AI can generate fake news & deepfakes

Bias & Ethics - AI can reflect biases in its training data

Creativity Debate - Is AI-generated content real creativity?

Data Privacy - AI models are trained on vast amounts of public data

The Future of Generative AI

More human-like AI assistants

Personalized AI-generated content for individuals

AI that co-creates with humans in art, music, and writing

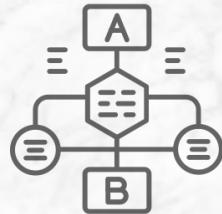
Ethical guidelines for responsible AI use



What is an LLM?

Definition: A Large Language Model (LLM) is an AI system trained on massive text data to understand and generate human-like language

Think of it like: A supercharged autocomplete that can write essays, answer questions, and even generate code!



How Do LLMs Work?

Training on Big Data → Trained on books, websites, and documents

Learning Patterns → Identifies relationships between words

Generating Responses → Predicts the next words based on context

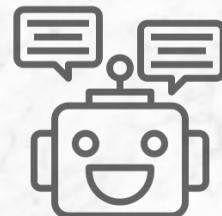
Where Are LLMs Used?

Chatbots & Virtual Assistants →  Siri, ChatGPT, Google Assistant

Content Creation →  Blog writing, copywriting, storytelling

Code Generation →  Assisting developers (GitHub Copilot)

Data Analysis →  Summarizing reports & extracting insights



Challenges & Ethical Concerns

Bias in AI →  LLMs learn from biased data

Misinformation →  They might generate incorrect or misleading answers

Privacy & Security →  Handling sensitive data responsibly is critical

The Future of LLMs

More accurate, faster, and multimodal AI (text + images + audio)

AI that reasons instead of just predicting words

Personalized AI models trained on user-specific data



 @ankitrathi



**Understanding
LLMs**



What is Agentic AI?

AI systems that act autonomously, making decisions, setting goals, and taking actions without constant human intervention
Like a self-driving car that plans its route, adapts to traffic, and makes real-time decisions all by itself



Key Features of Agentic AI

- Autonomous Decision-Making** - sets its own tasks and goals
- Planning & Reasoning** - doesn't just respond; it strategizes
- Adaptability & Learning** - improves based on feedback
- Memory & Context Awareness** - remembers past interactions
- Action Execution** - takes real-world actions, not just predictions

How Agentic AI Works?

Perception: observes the environment (data, sensors, user input)

Decision-Making: determines the best action based on goals

Action Execution: performs tasks autonomously

Feedback Loop: learns from successes and failures



Understanding

Agentic AI

in @ankitrathi

Traditional vs Agentic AI

Aspect	Traditional AI	Agentic AI
Task Execution	Predefined responses	Self-directed decision-making
Adaptability	Limited, follows rules	Learns and adapts
Autonomy	Requires human input	Acts independently
Memory	Short-term	Long-term memory & context



Challenges & Risks of Agentic AI

Loss of Control - AI taking actions beyond human oversight

Ethical Concerns - Who is responsible for AI decisions?

Unintended Consequences - AI optimizing for unintended goals

Safety & Security - Preventing rogue AI behaviour

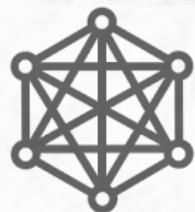


Understanding



Data Mesh

in @ankitrathi



What is Data Mesh?

a decentralized approach to data architecture

Moves away from centralized data lakes to domain-driven, self-service data ownership

Instead of one giant warehouse, each team has its own organized data store

Why Data Mesh? (Benefits)

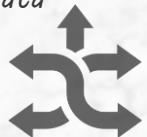


Scalability - No central team bottleneck

Faster Insights - Teams access the data they need without delays

Ownership & Quality - Teams take responsibility for reliable, high-quality data

Flexibility - Works with data lakes, warehouses, and real-time processing



Core Principles of Data Mesh

Domain-Oriented Ownership - Teams own & manage their data as a product

Data as a Product - Treat data like a service with defined consumers & quality standards

Self-Serve Infra - Empower teams to store, process, & share data independently

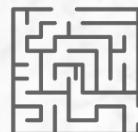
Federated Governance - Enforce global security, privacy, and standards

How Data Mesh Works

Each business unit (Finance, Marketing, HR, etc.) manages its own data

Data is discoverable, shareable, and reusable across teams

A common platform ensures security & interoperability without central bottlenecks



Challenges of Data Mesh

Cultural Shift - Teams must take ownership of data

Standardization Needed - Common governance rules must be enforced

Tech Complexity - Requires the right tools for seamless self-service

What is Data Quality?

Data Quality measures how accurate, reliable, and useful data is for decision-making

Goal: Ensure data is fit for use—complete, consistent, and free from errors.
Like clean water for drinking—bad data leads to bad decisions!



Why Does Data Quality Matter?

Better Decisions - Reliable data leads to accurate insights

Fewer Errors - Reduces costly mistakes in business & AI models

Compliance & Security - Ensures regulatory compliance (GDPR, HIPAA)

Higher Efficiency - Saves time spent fixing bad data

6 Key Dimensions of Data Quality

Accuracy - Data correctly represents real-world facts

Completeness - No missing or incomplete values

Consistency - Same data across different systems should match

Timeliness - Data is up-to-date and available when needed

Validity - Data follows rules & formats (e.g., correct date formats)

Uniqueness - No duplicate or redundant records



Understanding

Data Quality



[in](#) @ankitrathi

How to Improve Data Quality?

Data Validation - Check for errors before storing data

Deduplication - Remove duplicate records

Standardization - Enforce consistent formats and naming conventions

Automated Monitoring - Use tools to detect anomalies

Data Governance - Clear ownership & accountability for data

Challenges in Maintaining Data Quality

Human Errors - Manual data entry mistakes.



Data Silos - Inconsistent data across departments

Outdated Data - Old, irrelevant data reducing accuracy



Scaling Issues - Maintaining quality as data volume grows

What is Data Engineering?

It is the process of designing, building, and maintaining the systems that collect, store, and process data

Goal: Ensure data is accessible, reliable, and ready for analysis & AI

Like plumbing for data—moving and cleaning data so it's ready for use



Why is Data Engineering Important?



Reliable Data - Ensures accurate, well-structured data for analysis & AI

Scalability - Handles large-scale data efficiently

Faster Insights - Automates data flow for real-time analytics

Foundation for AI - AI & ML models rely on well-prepared data

Key Components of Data Engineering

Data Collection - Extracting data from sources (APIs, databases, logs)

Data Storage - Storing data in Data Lakes, Warehouses, or Lakehouses

Data Processing - Transforming raw data using ETL (Extract, Transform, Load) / ELT

Data Pipelines - Automating data flow using batch & real-time processing

Data Quality & Governance - Ensuring accuracy, security, and compliance



Tools & Technologies

Storage: Snowflake, BigQuery, Amazon S3, Delta Lake

Processing: Apache Spark, Databricks, dbt, Airflow

Pipelines: Kafka, Flink, Fivetran

Orchestration: Airflow, Prefect, Dagster

Challenges in Data Engineering

Data Silos - Breaking barriers between isolated data sources

Data Quality - Ensuring clean, consistent data

Real-Time Processing - Managing speed & reliability

Cost & Complexity - Scaling infrastructure efficiently



Understanding

in @ankitrathi

Data Engineering

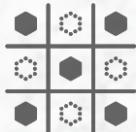
What is Data Analysis?

Process of cleaning, transforming, and interpreting data

To find meaningful patterns, trends, and insights

Goal: Convert raw data into useful knowledge for decision-making

Like solving a puzzle—each data point is a piece that helps complete the big picture



Why is Data Analysis Important?

Better Decision-Making - Data-driven insights lead to smarter choices

Problem-Solving - Identifies inefficiencies, risks, and opportunities

Predicting Trends - Helps businesses prepare for future changes

Competitive Advantage - Effective data analysis outperform others

Types of Data Analysis

Descriptive Analysis - "What happened?" (sales reports, trend charts)

Diagnostic Analysis - "Why did it happen?" (correlation, root cause analysis)

Predictive Analysis - "What might happen?" (forecasting, machine learning)

Prescriptive Analysis - "What should we do?" (decision-making models)

Understanding



Data Analysis

@ankitrathi



Common Data Analysis Techniques

Statistical Analysis - Mean, median, variance, hypothesis testing

Data Visualization - Charts, graphs, heatmaps for easy understanding

Correlation & Regression - Finding relationships between variables

Machine Learning Models - AI-driven pattern recognition

Text Analysis - Extracting insights from words and language

Challenges in Data Analysis

Dirty Data - Incomplete, inconsistent, or incorrect data

Data Overload - Too much data without clear focus

Bias & Misinterpretation - Drawing incorrect conclusions

Lack of Skills & Tools - Not everyone is trained in data analysis



Data Engineer ~ The Builder



What They Do? Build data pipelines & manage storage
Key Skills: SQL, Python, ETL, Cloud, Big Data
Challenges: Dirty data, pipeline failures, scalability
Future Trends: Real-time streaming data, Data Mesh, AI-powered data engineering

Data Analyst ~ The Storyteller

What They Do? Analyze data, create dashboards & reports

Key Skills: SQL, Excel, Tableau, Python, Business Acumen

Challenges: Messy data, unclear business questions, ad-hoc requests

Future Trends: Self-service analytics, AI-powered BI tools, Automated reporting



Top 5

Data & AI Roles

in @ankitrathi

Data Scientist ~ The Predictor



What They Do? Build ML models & derive patterns

Key Skills: Python, ML/DL, Statistics, AI Ethics

Challenges: Model deployment, bias, explainability

Future Trends: AI explainability, Edge AI, Ethical AI & regulation

AI/ML Engineer ~ The Deployer

What They Do? Deploy, monitor & optimize ML models

Key Skills: TensorFlow, Docker, MLOps, Cloud AI

Challenges: Model drift, latency, security

Future Trends: Low-latency AI, AI-powered DevOps, Federated Learning



Data/AI Product Manager ~ The Strategist

What They Do? Bridge business & AI, drive AI adoption

Key Skills: AI Strategy, Product Management, Communication

Challenges: AI ROI, adoption resistance, ethical concerns

Future Trends: AI-driven decision-making, AI governance & compliance, No-code AI platforms

Why is Data Called the “New Oil”?

Like oil in the Industrial Age ~ data is the key resource in the Digital Age

Raw data has no value until processed & refined—just like crude oil

AI & Analytics are the engines that extract value from data

Data is the new oil, AI is the refinery, and insights are the fuel powering businesses

How Businesses Leverage Data & AI

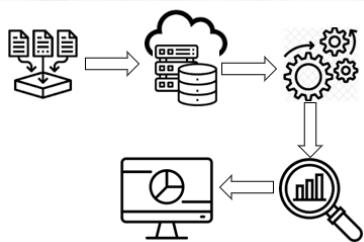
Personalization - Netflix, Amazon, Spotify use AI to recommend content & products

Data-Driven Decisions - Companies like Google & Tesla optimize strategies using data insights

Automation & AI - Chatbots, fraud detection, and predictive maintenance

Monetization - Tech giants sell data-driven advertising & insights (Google, Facebook)

The Data & AI Value Chain



Data Collection - Sensors, IoT, social media, transactions

Storage & Processing - Data lakes, warehouses, cloud computing

AI & Analytics - Machine learning, deep learning, business intelligence

Actionable Insights - Dashboards, reports, predictions

Business Impact - Cost savings, revenue growth, innovation

The Future of Data & AI

AI-Powered Everything - AI assistants, automation, autonomous systems

Real-Time Decision Making - Edge computing & AI-driven analytics

Responsible AI & Ethics - Transparency, fairness, and reducing bias

Data Privacy & Security - Regulations like GDPR & AI governance



Challenges & Risks



Data Privacy Issues - Who owns your data?



Bias in AI - Unfair outcomes due to biased training data

Scalability - Managing the explosion of global data



Ethical Concerns - Deepfakes, misinformation, surveillance risks

The Rise of

Data & AI



What is AI?

simulation of human intelligence in machines

Learning - Adapts from data

Reasoning - Makes decisions

Self-correction - Improves over time



Types of AI

Narrow AI (Weak AI) → Specialized in one task (Siri, Google Translate)

General AI (Strong AI) → Thinks like a human (still theoretical)

Super AI → More intelligent than humans (future concept)

AI Subfields

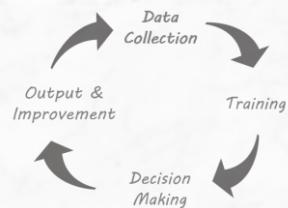
Machine Learning (ML) - Learns from data (Netflix recommendations)

Deep Learning (DL) - AI mimicking the human brain (self-driving cars)

Natural Language Processing (NLP) - Understands human language (Chatbots)

Computer Vision - Recognizes images (Face recognition)

How AI Works



Data Collection - AI learns from massive datasets

Training - Models adjust through experience

Decision Making - AI analyzes patterns

Output & Improvement - AI refines predictions over time

AI in Everyday Life

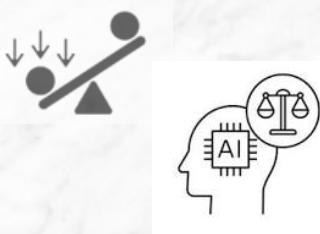
Voice Assistants (Alexa, Google Assistant)

Recommendation Systems (Netflix, YouTube)

Healthcare (Disease diagnosis, robotic surgery)

Autonomous Vehicles (Self-driving cars)

Finance & Security (Fraud detection, stock predictions)



AI Challenges & Ethics

Bias in AI - Unfair outcomes due to biased data

Privacy Issues - AI tracking and surveillance concerns

Job Automation - AI replacing jobs

Ethical AI - Ensuring AI benefits society



Understanding

AI

in @ankitrathi

Understanding

Data

in @ankitrathi



What is Data?

Its raw information in the form of numbers, text, images, or symbols



Data Formats

Structured (spreadsheets, databases)

Unstructured (emails, videos, social media posts)

Semi-structured (JSON, XML)

Data Processing Cycle

Collection - Sensors, surveys, transactions

Storage - Databases, cloud, servers

Processing - Sorting, filtering, analysing

Analysis - Trends, patterns, insights

Visualization - Graphs, charts, dashboards



Data Types & Examples

Quantitative (Numbers) → Sales figures, temperature

Qualitative (Descriptions) → Customer reviews, comments

Big Data (Massive sets) → Social media trends, IoT sensor data

Importance of Data

Better Decisions - Business strategies, healthcare, AI

Efficiency - Automation, predictive models

Innovation - Machine learning, scientific research



Data Challenges

Data Privacy & Security - Hacks, leaks, GDPR

Data Overload - Too much data, hard to analyze

Bias & Accuracy - Incorrect or misleading data

Analogy with a Car

Data Strategy → GPS & Roadmap

(Defines direction and purpose)

Ensures the organization knows where it's going with data, aligning it with business goals

Data Architecture → Accelerator

(Speeds things up)

Provides a structured framework that enables fast and scalable data processing

Data Governance → Brakes

(Ensures control and compliance)

Puts guardrails in place to ensure data quality, security, and compliance

Data Engineering → Fuel & Transmission

(Moves data efficiently)

Builds and maintains pipelines that deliver data where it's needed, ensuring smooth movement

Data Management → Engine Maintenance

(Keeps things running smoothly)

Ensures data is properly stored, processed, and maintained over time

Data Science → Turbocharger

(Adds power and intelligence)

Uses advanced models and algorithms to extract deeper insights and predictions.

Data Analytics → Dashboard & Gauges

(Provides insights to the driver)

Helps monitor performance, trends, and issues to make informed decisions

Understanding

Data & AI

 @ankitrathi

Ecosystem