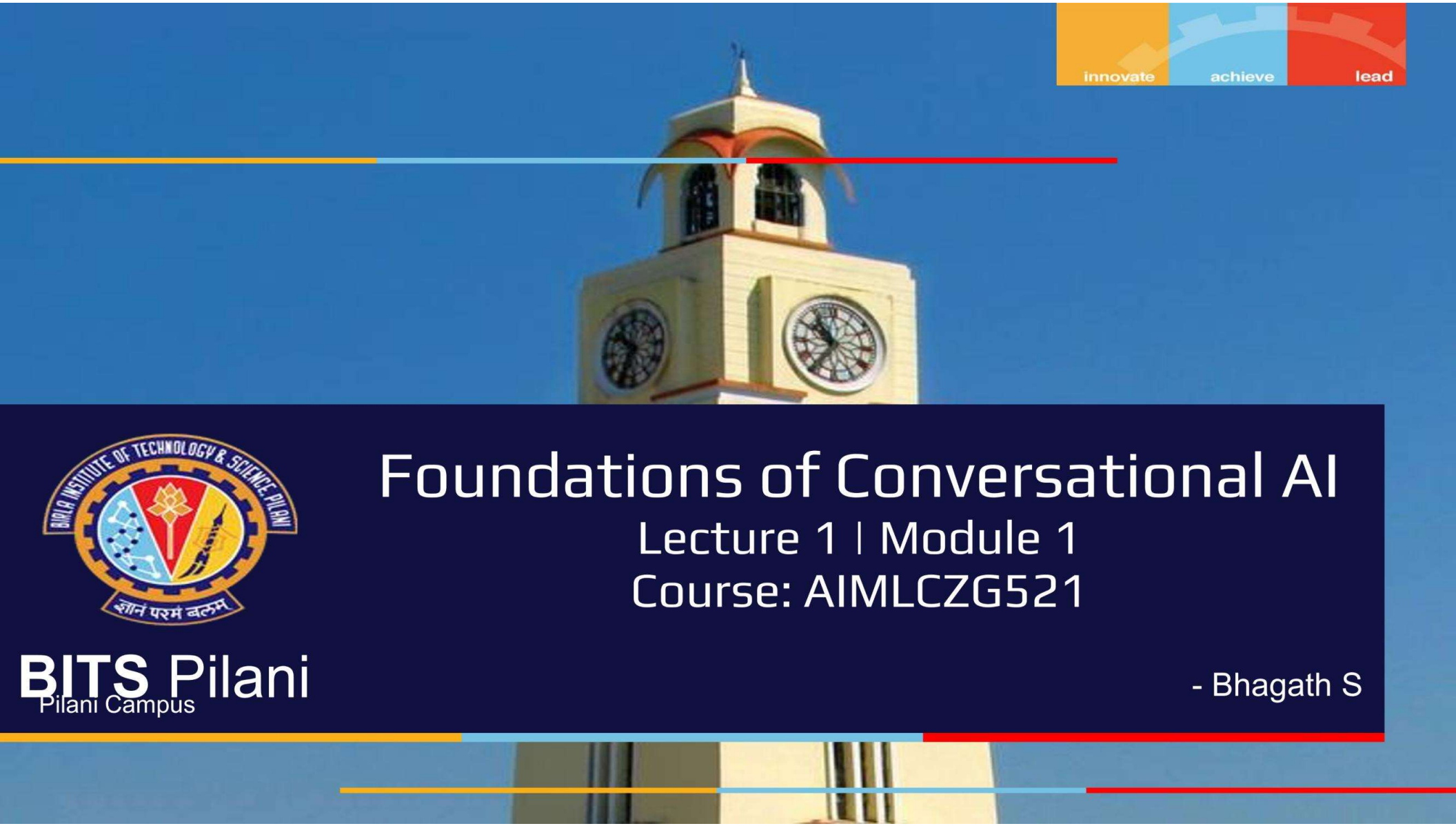


innovate achieve lead

# Foundations of Conversational AI

## Lecture 1 | Module 1

## Course: AIMLCZG521

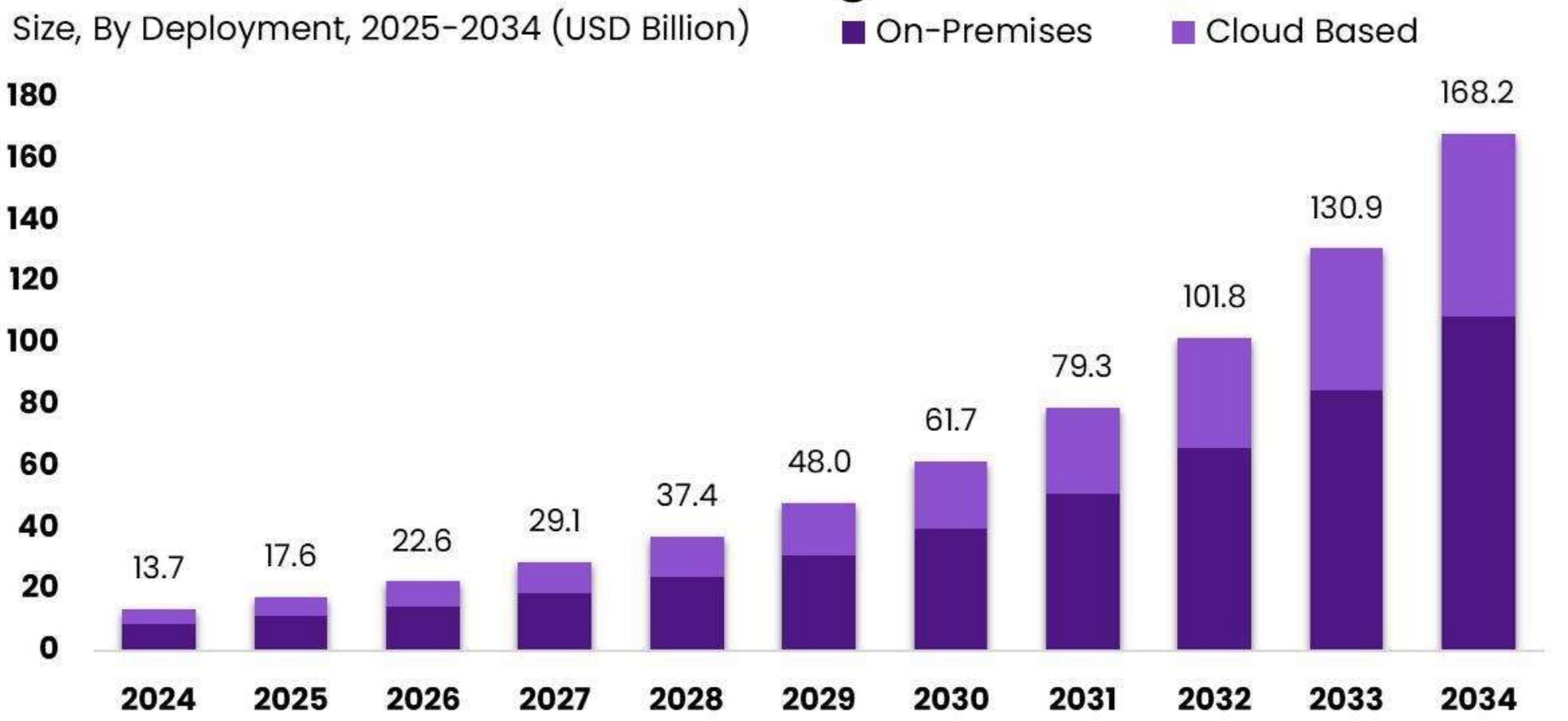
- Bhagath S

BITS Pilani

Pilani Campus

# Conversational AI & Virtual Agents Market

Size, By Deployment, 2025-2034 (USD Billion)

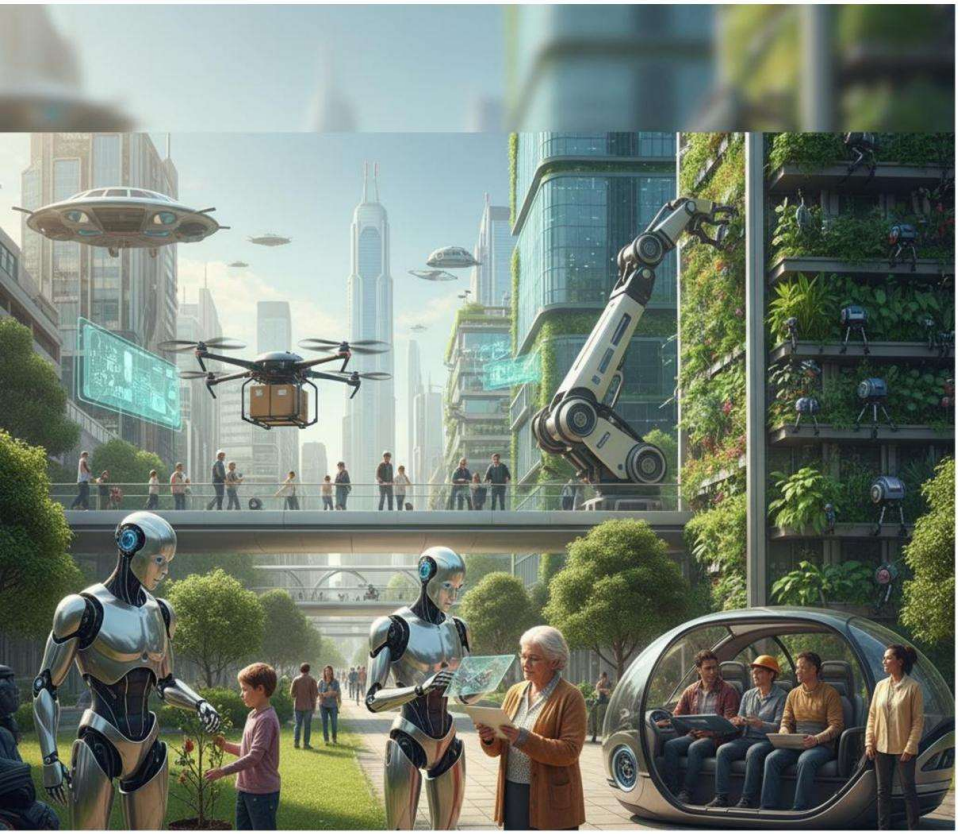



**The Market will Grow At the CAGR of: 28.5%**

**The Forecasted Market Size for 2034 in USD: $168.2B**

market.us
ONE STOP SHOP FOR THE REPORTS

# What is Conversational AI?



Conversational AI refers to technologies that enable computers to understand, process, and respond to human language in natural, human-like ways through spoken or written interactions.

## Core Capabilities

- **Natural Language Understanding (NLU)**
  Understanding user intent and entities
- **Dialogue Management**
  Managing conversation flow and context
- **Natural Language Generation (NLG)**
  Generating human-like responses
- **Context Awareness**
  Remembering conversation history
- **Multi-turn Interaction**
  Handling complex dialogues
- **Personalization**
  Adapting to user preferences

## Examples in Daily Life

**Voice Assistants:** Alexa, Siri, Google Assistant

**Customer Support:** Banking chatbots, e-commerce help

**Virtual Agents:** Healthcare triage, HR assistants

# The Evolution Journey (1960s - 2025)

| Era | Technology | Capabilities | Limitations |
|---|---|---|---|
| **1960s-1990s** Rule-Based | Pattern matching, keyword detection (ELIZA, ALICE) | Simple Q&A, scripted responses | No learning, rigid, no context |
| **2000-2010** Statistical ML | SVMs, CRFs, HMMs for NLU/NER | Intent classification, entity extraction | Limited context, hand-crafted features |
| **2010-2017** Deep Learning | RNNs, LSTMs, Seq2Seq, Attention | End-to-end learning, better context | Data hungry, task-specific training |
| **2017-2020** Transformers | BERT, GPT-2, T5 (pre-trained models) | Transfer learning, contextual embeddings | Still task-specific fine-tuning needed |
| **2020-2023** LLMs & Gen AI | GPT-3, ChatGPT, Claude, PaLM | Few-shot learning, general purpose, fluent generation | Hallucinations, no real-time data, no actions |
| **2023-2025** Agentic AI | LLMs + Tools + Memory + Planning | Execute actions, multi-step reasoning, tool use, autonomous workflows | Complex orchestration, scaling challenges, cost optimization needed |

# Evolution Deep Dive: Early Stages

## 1. Rule-Based Systems (1960s-1990s)

**Example: ELIZA (1966)**

```
IF user_input contains "mother": RESPOND "Tell me more about your family" IF user_input contains
"sad": RESPOND "Why do you feel sad?"
```

**Limitations:** No learning, brittle, couldn't handle variations, no real understanding

# Evolution Deep Dive: Early Stages

## 2. Statistical ML Era (2000-2010)

**Key Techniques:**

- **Intent Classification:** SVM, Naive Bayes to classify user intent
- **Named Entity Recognition:** CRF, HMM for extracting entities
- **Dialogue State Tracking:** Markov models for conversation flow

**Example Framework:** Microsoft LUIS, IBM Watson (early versions)

**Advance:** Could learn from data, handle variations better

**Limitations:** Required feature engineering, limited context understanding

# Evolution Deep Dive: Deep Learning Era

## 3. Deep Learning Revolution (2010-2017)

**Key Architectures:**

- **RNNs/LSTMs:** For sequence modeling, context retention
- **Seq2Seq Models:** Encoder-decoder for response generation
- **Attention Mechanism:** Focus on relevant parts of input

**Example Frameworks:** Rasa (open source), Google Dialogflow

**Advance:** End-to-end learning, better context, no hand-crafted features

**Limitations:** Data hungry, task-specific models, limited transfer learning

# Evolution Deep Dive: Deep Learning Era

## 4. Transformer Era (2017-2020)

**Breakthrough: "Attention is All You Need" (2017)**

- **Pre-trained Models:** BERT, GPT-2, T5, RoBERTa
- **Transfer Learning:** Pre-train on massive data, fine-tune for tasks
- **Contextual Embeddings:** Word meaning depends on context

**Impact on Conversational AI:**

- Better intent classification with fewer examples
- Improved entity recognition
- More natural response generation

# Evolution: Modern Era (2020-2025)

innovate achieve lead

## 5. Large Language Models (2020-2023)

**Game Changers: GPT-3 (2020), ChatGPT (2022), Claude, PaLM**

- **Massive Scale:** Billions to trillions of parameters
- **Few-Shot Learning:** Learn tasks from examples in prompt
- **General Purpose:** One model for many tasks
- **Human-like Fluency:** Natural, contextual responses

**Revolution for Conversational AI:**

• No fine-tuning needed for many tasks

• Understand complex, multi-turn conversations

• Generate creative, contextual responses

**But still limited:** No real-time data, can't take actions, hallucinations

# Evolution: Modern Era (2020-2025)

## 6. Agentic Conversational AI (2023-2025)

**The Next Frontier: LLMs + Tools + Memory + Planning**

- **Tool Use:** Can call APIs, search databases, execute code
- **Multi-step Reasoning:** Break down complex tasks
- **Memory Systems:** Remember user preferences, conversation history
- **Autonomous Actions:** Book appointments, send emails, create documents

**This is what we'll build in this course!**

# Architecture Evolution: Then vs Now

## Traditional Architecture (Pre-2020)

```
User Input ↓ Speech Recognition (if voice) ↓
Natural Language Understanding • Intent
Classification • Entity Extraction ↓ Dialogue
Manager • State Tracking • Policy (rules/ML) ↓
Natural Language Generation • Template-based •
Rule-based ↓ Text-to-Speech (if voice) ↓
Response
```

**Frameworks:** Rasa, Dialogflow, Microsoft Bot Framework, Amazon Lex

## Modern Agentic Architecture (2023+)

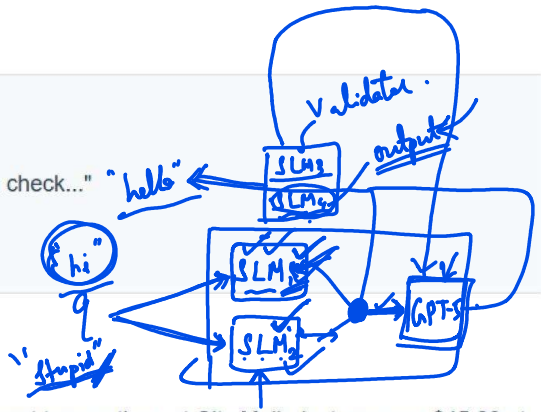
```
User Input ↓ LLM-based Understanding • Intent +
Entities in one pass ↓ Agentic Orchestration
Layer • Planning (break down task) • Tool
Selection • Memory Retrieval ↓ Tool Invocation
• API calls • Database queries • Code execution
↓ Memory Update • Store context • Update user
profile ↓ LLM-based Generation • Contextual
response ↓ Safety & Validation ↓ Response
```

**Frameworks:** LangChain, LlamaIndex, AutoGPT, CrewAI

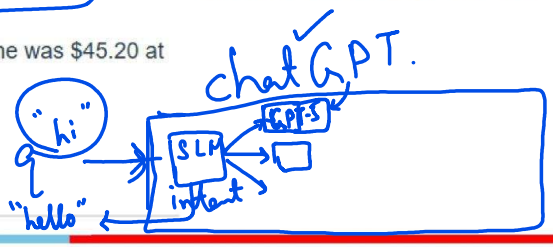
# Use Case: Customer Support Evolution

## Scenario: Banking Customer Support

| Approach | Capabilities | User Experience |
|---|---|---|
| **Rule-Based (2000s)** | • FAQ matching<br>• Keyword detection<br>• Fixed responses | User: "I lost my card"<br>Bot: "For lost card, press 1. For stolen, press 2"<br>**Rigid, frustrating** |
| **Intent-Based (2015)** | • Intent classification<br>• Entity extraction<br>• Dialogue flow | User: "I lost my card"<br>Bot: "I'll help you block your card. Which card - Credit or Debit?"<br>**Better, but limited** |
| **LLM-based (2023)** | • Natural conversation<br>• Context understanding<br>• Fluent responses | User: "I think I left my card at the restaurant"<br>Bot: "I understand. Let me help you block it temporarily while you check..."<br>**Natural, but no action** |
| **Agentic AI (2025)** | • Everything above +<br>• Access banking system<br>• Execute transactions<br>• Multi-step actions | User: "I lost my card at City Mall"<br>Bot: "I've immediately blocked your card ending in 1234. I see recent transactions at City Mall - last one was $45.20 at Starbucks 2 hours ago. Should I order a replacement card to your home address?"<br>**Proactive, action-oriented** |




**Business Impact:** Resolution time reduced from 15 minutes (human agent) → 2 minutes (agentic AI). Customer satisfaction increased by 40%.



# Components of Modern Conversational AI

### 1. Natural Language Understanding

**Role:** Understand what user wants

- Intent classification
- Entity extraction
- Sentiment analysis
- Context understanding

**Modern approach:** LLM-based, single pass

### 2. Dialogue Management

**Role:** Manage conversation flow

- State tracking
- Context maintenance
- Turn-taking
- Error handling

**Modern approach:** LLM reasoning + Memory systems

# Components of Modern Conversational AI

### 3. Knowledge Access

**Role:** Retrieve relevant information

- Vector databases
- Semantic search
- RAG (Retrieval-Augmented Generation)
- Real-time data access

**Modern approach:** Embeddings + Hybrid retrieval

### 4. Action Execution

**Role:** Take actions on behalf of user

- Tool/function calling
- API integrations
- Database operations
- External service invocation

**Modern approach:** Agentic tool use

# Components of Modern Conversational AI

### 5. Response Generation

**Role:** Generate natural responses

- Contextual generation
- Personality/tone
- Multi-modal output
- Structured responses

**Modern approach:** LLM generation with control

### 6. Memory Systems

**Role:** Remember user context

- Short-term (conversation)
- Long-term (user profile)
- Episodic memory
- Semantic memory

**Modern approach:** Vector + SQL hybrid

# Framework Evolution for Conversational AI

## Traditional Frameworks (2015-2020)

| Framework | Approach | Use Case |
|---|---|---|
| **Rasa** | Intent + Entity + Dialogue Policies | Custom chatbots, on-premise deployment |
| **Google Dialogflow** | Intent-based, ML-powered NLU | Voice assistants, Google integrations |
| **Microsoft Bot Framework** | Intent + Entity + Dialogue Management | Enterprise bots, Azure integrations |
| **Amazon Lex** | Intent-based, Alexa-powered | Voice + text bots, AWS services |

**Key Shift:** From intent-based dialogue systems to LLM-powered agentic systems with tool use and planning capabilities.

# Framework Evolution for Conversational AI

## Traditional Frameworks (2015-2020)

**Key Shift:** From intent-based dialogue systems to LLM-powered agentic systems with tool use and planning capabilities.

## Modern Agentic Frameworks (2023-2025)

# Framework Evolution for Conversational AI

## Modern Agentic Frameworks (2023-2025)

| Framework | Approach | Use Case |
|---|---|---|
| **LangChain / LangGraph** | LLM orchestration + Tools + Memory + Workflows | Complex agents, RAG, multi-step reasoning, production systems |
| **LlamaIndex** | Data-centric, RAG-optimized pipelines | Document Q&A, knowledge bases, enterprise search |
| **Semantic Kernel** | Microsoft's SDK for AI orchestration | Enterprise, .NET/Azure ecosystem integration |
| **Haystack** | End-to-end NLP framework | Production RAG systems, search applications |
| **AutoGen** | Multi-agent conversation framework | Complex workflows with agent collaboration |

# Course Content

**The Paradigm Shift:** Conversational AI has fundamentally changed from rule-based systems to intelligent, autonomous agents. This course prepares you for the **current and future** state of conversational AI.

## Module 1: Foundations

- **Embeddings & Vector Search:** Modern NLU requires semantic understanding → vector embeddings enable finding relevant context and intents
- **Model Landscape:** Understanding LLMs (GPT-4, Claude, Gemini) that power modern conversational AI
- **Cost Engineering:** Real-world deployment requires optimizing token usage and model selection

## Module 2: Core Building Blocks

- **Function Calling:** How conversational agents execute actions (book flights, query databases)
- **Memory Systems:** How agents remember conversation history and user preferences
- **RAG Pipelines:** How agents access knowledge bases to answer questions accurately

# Course Content


### Module 3: Autonomous Agents

- **Planning:** How agents break down complex user requests into steps
- **Multi-Agent Systems:** How specialized agents collaborate (customer support routing)
- **Evaluation:** Measuring quality of conversational AI systems

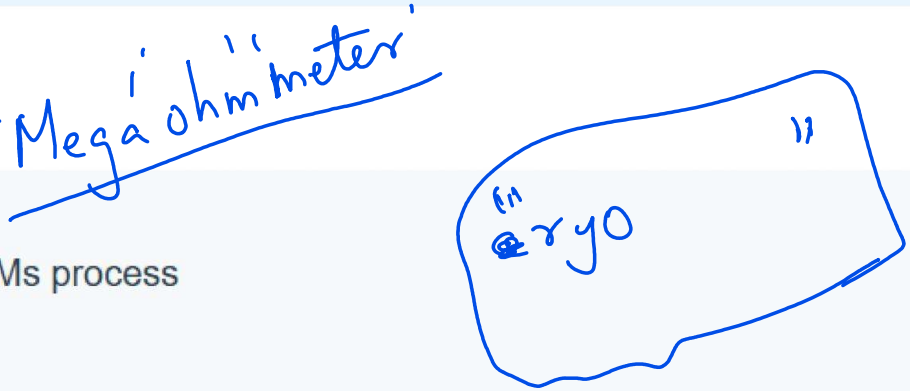
### Module 4: Production Ecosystem

- **Security:** Protecting against prompt injection, PII leakage in conversations
- **Protocols (MCP, A2A):** Standard ways for agents to communicate
- **Ethics:** Ensuring fair, unbiased conversational AI systems

# NLP Foundations for Conversational AI

Core NLP concepts that directly impact conversational agent design and performance

## 1. Tokenization - The Foundation




**What:** Breaking text into subword units (tokens) that LLMs process

**Why it matters for Conversational AI:**

- **Cost:** API pricing is per token → impacts conversation economics
- **Context Window:** Limits conversation length (e.g., 200K tokens = ~150K words)
- **Latency:** More tokens = slower response in conversations
- **Quality:** Tokenization affects understanding of domain-specific terms

# NLP Foundations for Conversational AI

## 2. Context Windows - Conversation Memory

**What:** Maximum text length (in tokens) an LLM can process at once

**Current State (2025):**

- GPT-4 Turbo: 128K tokens (~96K words) - standard production use
- Claude 3.5 Sonnet: 200K tokens (~150K words) - extended conversations
- Gemini 1.5 Pro: 1M tokens (~750K words) - entire codebases
- Emerging models: 2M+ tokens for specialized use cases

**Challenge:** "Lost in the middle" - models struggle with info in long contexts

**Solution:** RAG + Memory systems (covered in Module 2)

# Tokenization: How It Works

## Common Tokenization Algorithms

1. **Byte-Pair Encoding (BPE)** - Used by GPT models
2. **SentencePiece** - Used by many multilingual models
3. **WordPiece** - Used by BERT and similar models

## Example: How Text Becomes Tokens

| Text | Tokens | Count |
|---|---|---|
| "Hello World" | ["Hello", " World"] | 2 tokens |
| "artificial intelligence" | ["art", "ificial", " intelligence"] | 3 tokens |
| "GPT-4" | ["G", "PT", "-", "4"] | 4 tokens |
| "Book a flight to NYC" | ["Book", " a", " flight", " to", " NYC"] | 5 tokens |

# Tokenization: How It Works


## Impact on Conversational AI

**Scenario:** Customer support conversation (2025 typical costs)

- Average conversation: 40-60 turns (user + agent exchanges)
- Average tokens per turn: 15-25 tokens
- Total per conversation: ~800-1,200 tokens
- Cost with GPT-4o: ~$0.01-0.03 per conversation
- Cost with GPT-3.5 Turbo: ~$0.002-0.005 per conversation
- At 10,000 conversations/day with GPT-4o: $100-300/day

**Key insight:** Model selection and optimization can reduce costs by 10-20x. We'll cover these techniques in Lecture 11.

# Hands-On: Byte-Pair Encoding

innovate achieve lead

## Understanding Tokenization with Code

We'll explore how text is converted to tokens using the tiktoken library

## Demo Components:

- **Text to Tokens:** See how different texts are tokenized
- **Token Counting:** Calculate tokens for a sample conversation
- **Cost Analysis:** Understand the economic implications
- **Model Comparison:** Tokenization differences across models

# Transformers & LLMs: The Brain of Modern Conversational AI


You learned about **transformers** in your Deep Learning course. Let's connect that to **Conversational AI**.

## What Transformers Gave Us

### Technical Capabilities

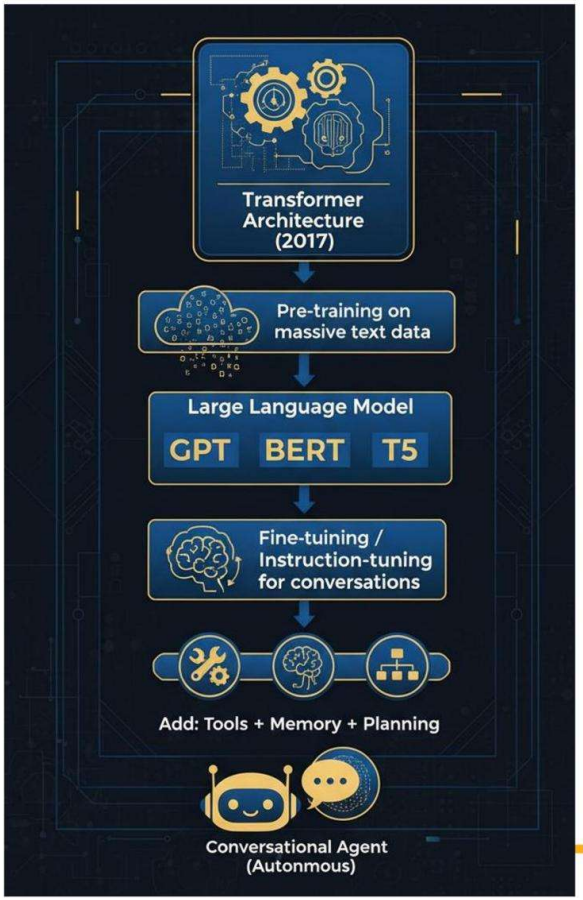
- **Self-Attention:** Understand relationships between words
- **Contextual Understanding:** Word meaning depends on context
- **Long-Range Dependencies:** Connect ideas across long text
- **Parallel Processing:** Faster than RNNs

### Impact on Conversations

- **Multi-turn coherence:** Remember earlier in conversation
- **Intent understanding:** Grasp user goals from context
- **Natural responses:** Generate human-like replies
- **Ambiguity handling:** Resolve unclear requests

# Transformers & LLMs: The Brain of Modern Conversational AI

## From Transformer → LLM → Conversational Agent




**Key Insight:** LLM is the "brain" but needs additional systems (tools, memory, planning) to become a fully functional conversational agent.

# LLM Capabilities & Limitations for Conversational AI


## ✅ What LLMs Do Well

- **Natural conversation:** Human-like dialogue
- **Intent understanding:** Grasp user goals
- **Context retention:** Multi-turn conversations
- **Response generation:** Fluent, contextual replies
- **Summarization:** Condense information
- **Entity extraction:** Identify key information
- **Sentiment analysis:** Understand user emotions

## ❌ LLM Limitations

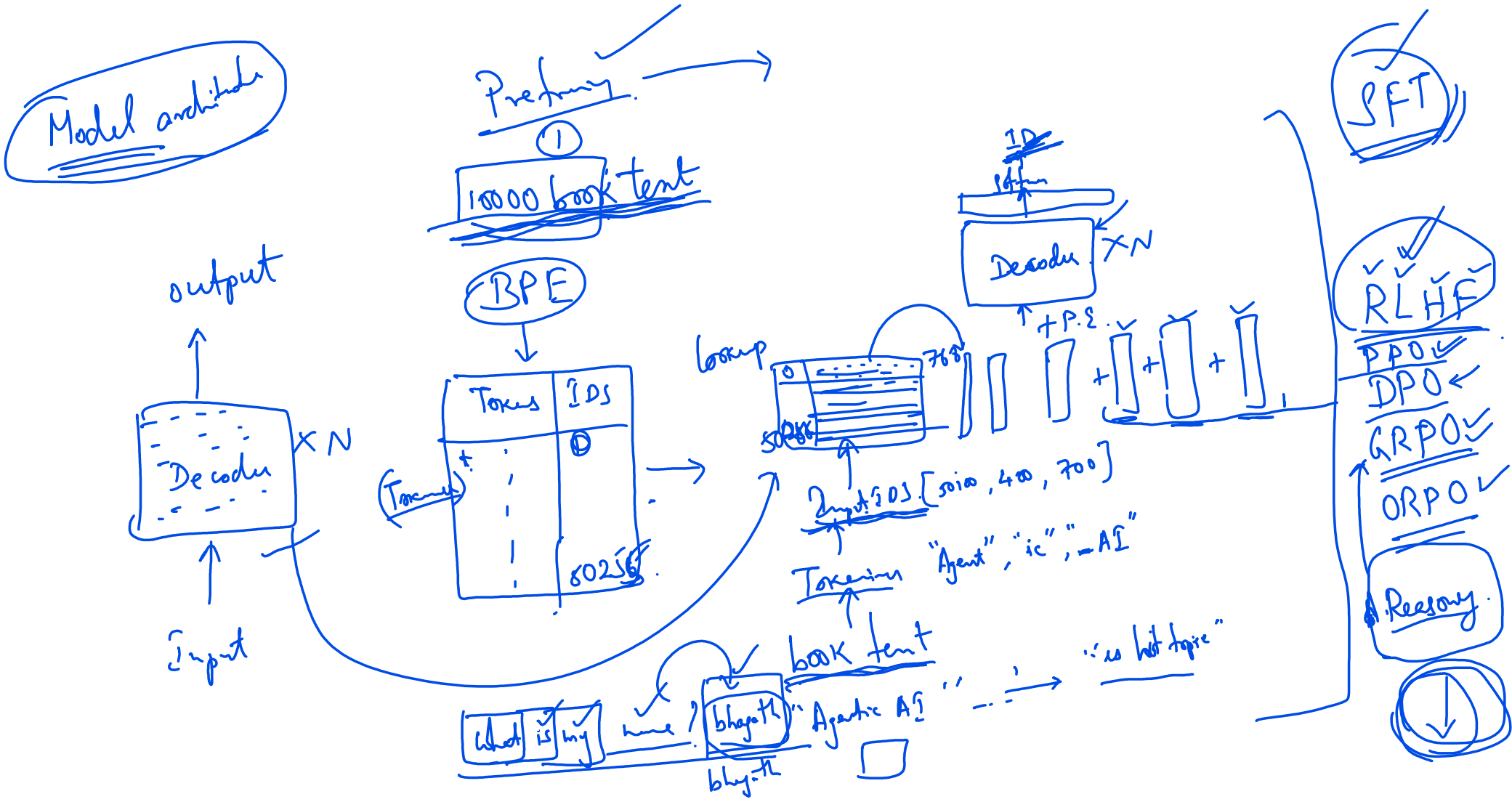
- **No real-time data:** Training cutoff date
- **Hallucinations:** Generate plausible but false info
- **Can't take actions:** No access to external systems
- **No memory:** Forget after context window
- **Calculation errors:** Struggle with math
- **Consistency:** Different answers to same question
- **No verification:** Can't check facts

# LLM Capabilities & Limitations for Conversational AI

## Why We Need Agentic Architecture

**Solution: Augment LLMs with external capabilities**

| LLM Limitation | Agentic Solution | Example |
|---|---|---|
| No real-time data | Tool calling (APIs, search) | Check current flight prices |
| Hallucinations | RAG (Retrieval-Augmented Generation) | Ground answers in company docs |
| Can't take actions | Function calling | Book appointment, send email |
| No memory | External memory systems | Remember user preferences |
| Calculation errors | Calculator tool, code execution | Precise math computations |


Model architecture
Pretraining
1
10000 book text
BPE
Tokens
IDs
50256
Tokenizer
output
Decoder
× N
Input
lookup
768
Input IDs [5010, 400, 700]
Tokenizer
"Agent", "ic", "-AI"
book text
what is my name ? bhagath
bhagath
Agentic AI
"is hot topic"
1D
Decoder
× N
+ P.E
SFT
RLHF
PPO
DPO
GRPO
ORPO
Reasoning

# Agentic Conversational AI: Request → Response Pipeline

How a modern conversational agent processes user input

| Step | Details |
|---|---|
| 📥 1. REQUEST | • User input received and validated • Parse user intent • Extract entities • Input sanitization (security) |
| 🔀 2. ROUTING | • Intent classification & task decomposition • Classify intent type • Determine required tools • Route to appropriate sub-agent |
| 🧠 3. REASONING | • Planning & decision making with LLM • Break down into steps • Identify information gaps • Plan tool call sequence |
| 🛠️ 4. TOOL INVOCATION | • External API calls & actions • Execute API calls • Database queries • Code execution |
| 💾 5. MEMORY | • Context storage & retrieval • Store conversation history • Update user preferences • Maintain session state |
| 🛡️ 6. SAFETY | • Guardrails & output validation • Check for toxic content • Verify factual accuracy • PII redaction |
| 📤 7. RESPONSE | • Final output delivered to user • Natural language generation • Format for channel (text/voice/UI) • Send to user |

# Agentic Conversational AI: Request → Response Pipeline

How a modern conversational agent processes user input

| Step | Details |
|---|---|
| 1. REQUEST | • User input received and validated<br>• Parse user intent<br>• Extract entities<br>• Input sanitization (security) |
| 2. ROUTING | • Intent classification & task decomposition<br>• Classify intent type<br>• Determine required tools<br>• Route to appropriate sub-agent |
| 3. REASONING | • Planning & decision making with LLM<br>• Break down into steps<br>• Identify information gaps<br>• Plan tool call sequence |
| 4. TOOL INVOCATION | • External API calls & actions<br>• Execute API calls<br>• Database queries<br>• Code execution |

| Step | Details |
|---|---|
| 5. MEMORY | • Context storage & retrieval<br>• Store conversation history<br>• Update user preferences<br>• Maintain session state |
| 6. SAFETY | • Guardrails & output validation<br>• Check for toxic content<br>• Verify factual accuracy<br>• PII redaction |
| 7. RESPONSE | • Final output delivered to user<br>• Natural language generation<br>• Format for channel (text/voice/UI)<br>• Send to user |

# Lifecycle Example: Banking Conversational Agent

**User Request: "What's my account balance and recent transactions?"**

| Stage | What Happens |
|---|---|
| **1. Request** | • Intent: account_inquiry<br>• Entities: {type: ["balance", "transactions"]}<br>• Validate: User authenticated |
| **2. Routing** | • Route to: Banking Agent<br>• Required tools: [get_balance, get_transactions]<br>• Permission check: User has access |
| **3. Reasoning** | • Plan:<br>1. Fetch account balance<br>2. Fetch recent transactions (last 5)<br>3. Format response with both |

| Stage | What Happens |
|---|---|
| **4. Tool Invocation** | • Call: get_balance(user_id) → Returns: $5,432.10<br>• Call: get_transactions(user_id, limit=5)<br>• Returns: [Transaction1, Transaction2, ...] |
| **5. Memory** | • Store: User asked about balance at 2:30 PM<br>• Update: User preference for transaction details<br>• Context: Maintain for follow-up questions |
| **6. Safety** | • Validate: Balance and transactions are for correct u<br>• Check: No PII exposure in logs<br>• Verify: Response contains only authorized info |

# Lifecycle Example: Banking Conversational Agent

**User Request: "What's my account balance and recent transactions?"**

| Stage | What Happens |
|---|---|
| **7. Response** | "Your current balance is $5,432.10. Here are your recent transactions:<br>1. Starbucks - $4.50 (Today)<br>2. Amazon - $89.99 (Yesterday)<br>3. Shell Gas - $45.00 (Yesterday)<br>4. Grocery Store - $123.45 (2 days ago)<br>5. Netflix - $15.99 (3 days ago)<br>Would you like more details on any of these?" |

# Exercise: Map the Agent Lifecycle

innovate achieve lead

**Scenario:**

**"Find me a good Italian restaurant near my office that's open tonight and make a reservation for 2 at 7 PM"**

## Map This Request to the 7-Stage Lifecycle

# Exercise: Map the Agent Lifecycle

**Scenario:**

**"Find me a good Italian restaurant near my office that's open tonight and make a reservation for 2 at 7 PM"**

| Stage | What Needs to Happen? |
|---|---|
| **1. Request** | What entities need to be extracted from user input? |
| **2. Routing** | Which tools or agents are needed for this task? |
| **3. Reasoning** | What's the step-by-step plan to accomplish this? |
| **4. Tool Invocation** | What specific API calls need to be made? |
| **5. Memory** | What information should be stored for future use? |
| **6. Safety** | What validations are required before taking action? |
| **7. Response** | How should the agent communicate with the user? |

# Building a Simple Conversational Agent

## Demonstration: Weather Information Agent

A practical example using native OpenAI API in Python

## Implementation Stages

1. **Baseline:** LLM without tools - sees limitations
2. **Tool Definition:** Define weather function schema
3. **Tool Selection:** LLM decides when to use tool
4. **Execution:** Call actual weather API
5. **Response Generation:** LLM creates natural answer

## Example Interaction

**User:** "What's the weather like in Mumbai?"

**Agent:**

1. Extracts location: "Mumbai"
2. Calls weather API
3. Receives: 32°C, Humid, Partly Cloudy
4. Responds: "It's currently 32°C in Mumbai with partly cloudy skies and high humidity."

# Protocol Landscape for Conversational Agents

As conversational agents become more prevalent, they need **standardized ways to communicate** with tools, data, each other, and UIs.

## Why Protocols Matter

### Without Standards

- Every vendor has custom API
- Agents can't interoperate
- Vendor lock-in
- Duplicated integration effort

### With Standards

- Plug-and-play integrations
- Multi-agent collaboration
- Vendor flexibility
- Ecosystem growth

# Protocol Landscape for Conversational Agents

## The Protocol Landscape (2025)

| Protocol | Purpose | Use Case |
|---|---|---|
| **MCP** (Model Context Protocol) | Standardize how LLMs connect to data sources and tools | Agent accessing databases, files, APIs consistently |
| **OpenAI Assistant API** | Built-in tools, file access, code interpreter | Rapid agent development with OpenAI infrastructure |
| **LangGraph Protocol** | Graph-based agent workflows and state management | Complex multi-step reasoning, agent coordination |
| **Custom REST/GraphQL APIs** | Traditional service integration | Enterprise systems, legacy integrations |

**Note:** The protocol landscape is rapidly evolving. Standards like MCP are emerging, while many production systems still use custom APIs. We'll explore these in detail in Lectures 13-14.

# Production Concerns for Conversational AI

Building conversational agents that work in development is one thing. Building them to work **reliably at scale** in production is another.

### 1. Observability

**What to track:**

- Conversation flows
- Tool invocations
- Token usage per conversation
- Response latencies
- Error traces

**Tools:** LangSmith, Arize Phoenix, OpenTelemetry

### 2. Cost Management

**Optimization strategies:**

- Prompt caching (50-90% cost reduction)
- Model routing (use smaller models when appropriate)
- Token budgets per user/session
- Efficient retrieval (reduce context size)

**Example:** A customer support conversation can cost $0.02-0.10 depending on optimization

# Production Concerns for Conversational AI

### 3. Latency Budgets

**Target latencies:**

- Chat responses: < 2 seconds
- Tool execution: < 5 seconds
- Complex reasoning: < 30 seconds

**Key:** Set user expectations with progress indicators

### 4. Safety & Security

**Defense layers:**

- Input validation (prompt injection defense)
- PII detection and redaction
- Output filtering (toxicity, hallucinations)
- Human-in-the-loop for critical actions

**Principle:** Never rely on a single safety layer

**These must be considered from day one—we'll cover each in detail in Lectures 11-12**

# Key Takeaways

### 1. Evolution of Conversational AI

From rule-based (1960s) → ML (2000s) → Deep Learning (2010s) → Transformers (2017) → LLMs (2020) → Agentic AI (2023+)

### 2. Architecture Transformation

Traditional: Intent → Dialogue Manager → Template Response

Modern: LLM + Tools + Memory + Planning + Safety

### 3. Components Matter

Modern systems need: NLU, Dialogue Management, Knowledge Access, Action Execution, Response Generation, Memory

### 4. LLMs are the Brain

But they need external systems for tools, memory, real-time data, and action execution

### 5. Agent Lifecycle

Request → Routing → Reasoning → Tool Invocation → Memory → Safety → Response

### 6. Production is Different

Observability, cost optimization, latency management, and safety are non-negotiable

# Key Takeaways


### The Foundation is Set!

You now understand what Conversational AI is, how it evolved, and the architecture of modern agentic systems.

**Next:** We'll cover these in detail, starting with embeddings and vector search.

# Coming Up Next

## Lecture 2: Embeddings, Vector Search & Hybrid Retrieval

### What We'll Cover

- Deep dive into embeddings and semantic search
- Vector database architecture (HNSW, ANN)
- BM25 + Dense retrieval + RRF (hybrid search)
- Building a practical retrieval system
- Evaluation metrics for retrieval

### How to Prepare

- **Pre-read:** "Dense Passage Retrieval for Open-Domain QA" (Karpukhin et al., 2020)
- **Set up:** Python environment with OpenAI/Anthropic API keys
- **Install:** Libraries: openai, chromadb, rank-bm25
- **Review:** Today's slides on Canvas

# Questions & Discussion

innovate achieve lead

### Topics for Discussion:

- Conversational AI use cases in your organization
- Evolution and current state of the field
- Technical challenges you foresee
- Clarifications on any concepts
- Course logistics and expectations

### Thank you!

See you in Lecture 2! 🚀