# Summary of topics

Saturday, 3 May 2025 11:37 PM

**Agent:**

- A control flow mechanism where the LLM itself decides the execution path
- Features bi-directional processing with potential cycles
- Allows for more dynamic decision-making capabilities

**Chaining:**

- Takes the output of one language model and feeds it as input to another
- Creates a composed sequence of LLM calls
- Follows a unidirectional flow (left to right)
- Example: Retrieval Augmented Generation (RAG)
  - First LM processes the original question
  - Question gets embedded and similar documents are retrieved
  - Original prompt is augmented with retrieved information
  - Final LM generates the answer based on enhanced context

**Router:**

- A specialized chain that uses an LLM to determine execution paths
- Leverages the LLM's reasoning ability to decide which branch to follow
- Can direct flow to different operations (e.g., database search vs. web search)
- Provides increased flexibility in building complex AI systems

**Key Distinction:**

- Chains: Linear, predetermined flow of operations
- Agents: Dynamic flow with decision-making capabilities and potential cycles

**Function Calls:**

- A feature allowing LLMs to invoke external tools/functions
- Process:
  - Developer provides function descriptions (title, purpose, arguments, return values)
  - LLM determines when function use is appropriate based on the query
  - LLM specifies which functions to call with what arguments
  - Backend executes the functions and returns results
- Can be implemented using decorators for simplified orchestration

**Agent Fundamental Design:**

- Simple algorithmic flow:
  1. Start with user query
  2. LLM decides if a tool is needed (API call, database query, etc.)
  3. If tool is needed, execute it with LLM-specified arguments
  4. Return tool results to the LLM
  5. LLM either uses another tool or provides final answer to user
- Creates a loop of tool use and reasoning

**RAG (Retrieval Augmented Generation):**

- Enhances LLM responses by incorporating external knowledge
- Process:
  1. Receive user query
  2. Retrieve relevant information from external knowledge base
  3. Augment prompt with retrieved context
  4. Generate response using both query and retrieved information
- Helps overcome knowledge limitations and hallucinations

**LLM.txt:**

- A pattern for interfacing with language models through structured text
- Enables complex interactions via simple text-based input/output
- Standardizes how applications communicate with language models

**ReAct in LangChain:**

- Combines reasoning and action in agent workflows
- Process:
  1. Reasoning: LLM thinks through the problem
  2. Action: Executes tools based on reasoning

2. Action: Executes tools based on reasoning
3. Observation: Processes results from actions
4. Repeats cycle until completion

- Creates more effective agents through structured thought-action patterns

**Pipfile:**

- Pipenv manages 2 files pipfile - declares **what** packages your project needs and **Pipfile.lock** – **locks down exact versions** of all packages (and sub-dependencies) that should be installed.
- Pipfile.lock is important:
  - Ensures Reproducible Environments
  - Protects Against Breaking Changes
  - Improves Install Performance
  - Safer Deployment

# MCP (Model Completion Protocol)

MCP is a structured framework for prompting language models with clear conventions for input/output formatting and function calling.

MCP follows specific conventions:

- Uses delimiters to separate different parts of the prompt
- Defines structured formats for system messages, user inputs, and model responses
- Creates standardized ways to specify tool use and function calls