Background / Motivation :
Previous research on Large Language Models (LLMs) has largely studied their **reasoning abilities** (e.g., Chain-of-Thought prompting) and **acting capabilities** (e.g., action plan generation) as separate topics.

Proposal:
This paper proposes **ReAct**, an approach that enables LLMs to generate both **reasoning traces and task-specific actions in an interleaved manner**, fostering greater synergy between the two.

Mechanism:
ReAct operates by allowing the LLM to alternate between **"Thought" (reasoning)** and **"Action" (acting)**.

- **Reasoning :** helps the model to induce, track, and update action plans, as well as handle exceptions
- **Actions :** the model to interface with and gather additional information from external sources
  - **Mitigates common LLM issues:**
    - **Combating Hallucination :** By interacting with external tools (e.g., a simple Wikipedia API), ReAct grounds its responses in retrieved factual information, significantly reducing the generation of fabricated content.
    - **Preventing Error Propagation :** External observations from actions allow the model to detect and correct errors in its reasoning process early on, preventing a single mistake from cascading through subsequent steps.

- **Result**
  - On **question answering (HotpotQA)** and **fact verification (Fever)**, ReAct successfully overcomes hallucination and error propagation by interacting with a Wikipedia API.
  - On two **interactive decision-making benchmarks (ALFWorld and WebShop)**, ReAct significantly outperforms imitation and reinforcement learning methods, achieving 34% and 10% higher absolute success rates, respectively, often with only one or two in-context examples.

★ how LLMs (specifically LangChain agents) can use various tools and APIs:

In LangChain, the agent can utilize a wide range of representative API/tool types, including:
Search APIs:
- **Google Search API / Bing Search API:** The most common and powerful tools. Used for searching for up-to-date information, news, and data from various websites.
- **Wikipedia API:** Specialized for searching the vast knowledge base of Wikipedia. (Used as an example in the ReAct paper.)
- **DuckDuckGo Search API:** A privacy-focused alternative to Google/Bing search APIs.
- **ArXiv API:** Used for searching scientific papers in the ArXiv database. Useful for research-related queries.

Calculator/Math APIs:
- **Python REPL (Read-Eval-Print Loop):** Executes Python code to perform complex mathematical calculations, data processing, and logical operations. Provides more accurate and complex computations than the LLM calculating directly.
- **Wolfram Alpha API:** A powerful computational knowledge engine that provides advanced mathematical calculations, scientific data, and statistical information.

Database/Vector Store APIs:
- **Pinecone, Weaviate, Chroma, FAISS, etc. (Vector Store APIs):** Store self-built knowledge bases (documents, FAQs, product descriptions, etc.) in vector form. They retrieve the most similar documents to a given query and provide them to the LLM. This is a core component of RAG (Retrieval-Augmented Generation).
- **SQL Database APIs:** Generate and execute SQL queries to retrieve or update information from relational databases.

General API Call / Web Request Tools:
- **RequestsWrapper:** Sends HTTP requests to specific website API endpoints to fetch information or perform actions (e.g., weather APIs, stock information APIs, currency exchange rate APIs).
- **Selenium/Playwright (Web Scraping):** Can be used to directly navigate websites and extract data. (More complex use cases.)

File System Interaction:
- **Read/Write File Tools:** Used to read from or write to files on the local file system (e.g., analyzing the content of a specific file, saving results to a file).

Code Execution Environments:
- **Jupyter/Colab Integration:** Allows for executing code cells and receiving results, enabling complex programming tasks like data analysis or graph generation.

Specific Service APIs:
- **Zapier NLA (Natural Language Actions):** Integrates with thousands of apps like Gmail, Slack, and Salesforce to perform real-world actions such as sending emails, sending messages, or updating CRM data.
- **Calendar API:** For querying and adding events.
- Type to enter text **Email API:** For sending and receiving emails.
- **E-commerce APIs:** For product search, order processing, etc. (The WebShop benchmark simulates such an environment.)

Core idea:

The LLM either learns the **functionality and input/output formats** provided by these APIs in advance or receives them via the prompt. The LLM then analyzes the user's request, determines which tool is needed, generates the appropriate arguments for that tool, and invokes it. The execution result from the tool is then fed back to the LLM to inform its next reasoning step or action.

Limitations/ points that mentioned by previous LLM research

Limitations of Chain-of-Thought (CoT):

CoT provides impressive step-by-step reasoning, but it's fundamentally an **isolated internal process**. It cannot **interact with or adapt to the external world**, nor can it **update its knowledge**. This leads to issues like **hallucination** and **error propagation**.

Limitations of Action-Oriented LLM Research:

While LLMs have been used for planning and acting in real-world environments (e.g., games, robotics) by converting observations to text and generating actions, **they often lack abstract reasoning for high-level goals and fail to maintain a working memory** to support complex sequential actions.

However, these methods lack the ability for the LLM to **abstractly "think" about high-level goals** or to **maintain a "working memory"** that remembers the results of previous actions and plans subsequent ones based on them (beyond simple reiteration of spatial facts).
In conclusion, this introduction argues that "to create an LLM that thinks and acts like a human, existing models that only 'think' (CoT) or only 'act' are insufficient. A new method is needed to tightly combine these two for synergistic effects!" It implicitly states that this is precisely the problem ReAct aims to solve.

★
ReAct's Operational Flow

three core elements—Thought, Action, and Observation—interact in a **repeated, interleaved sequence**.

The process unfolds follows :
★ **User Query → Thought → Action → Observation → Thought → Action → Observation → ... → Final Answer (Action: Finish)**

- **Dynamic Reasoning:** The LLM can continuously update and refine its 'Thought' based on real-time information obtained through 'Observation'.
- **External Information Utilization:** It overcomes the limitations of the LLM's internal knowledge by leveraging up-to-date information or specific domain knowledge from external sources.
- **Error Correction:** By analyzing 'Observation', the LLM can detect errors in its 'Action' or 'Thought' and correct them in subsequent steps.
- **Transparency:** The explicit 'Thought' steps make the LLM's decision-making process easily understandable to humans, showing how it arrived at an answer or performed an action.

Experiments

- **LLM Usage:** The research utilized a specific large language model (PaLM-540B) as its foundation.
- **Learning/Induction Method:** Instead of directly fine-tuning the model, it leveraged **few-shot in-context examples** provided within the prompt to induce the model to follow the ReAct's "Thought-Action-Observation" pattern. These examples were "human trajectories" – records of "thoughts, actions, and observations" generated by a human solving the task.
- **Controlled Frequency of Thought/Action Generation:**
- **For reasoning-centric tasks (e.g., QA, Fact Verification):** A rigid pattern was used where "Thought," "Action," and "Observation" appeared **alternatingly** and regularly.
- **For decision-making tasks (e.g., ALFWorld, WebShop):** The LLM was allowed to decide the **sparse** occurrence of "Thoughts," generating them only when deemed necessary. This demonstrates ReAct's flexible applicability.

the ReAct framework for knowledge-intensive reasoning tasks, specifically **multi-hop Question Answering (HotPotQA)** and **Fact Verification (FEVER)**.

Key Objective:
ReAct aims to leverage a Wikipedia API to retrieve information, demonstrating a **synergy between reasoning and acting**. The LLM uses reasoning to decide what to retrieve next (reason to act) and the retrieved information supports further reasoning (act to reason).

Question-Only:
- **Datasets:** HotPotQA (multi-hop QA) and FEVER (fact verification) are used.
- **Crucial Setup:** Models receive only the question/claim as input, with **no supporting paragraphs provided**.
- **Purpose:** This forces the model to **actively explore the external environment (Wikipedia API)** to gather necessary information, making ReAct's "Action" capability essential for problem-solving, rather than relying solely on internal knowledge.

★ **Designed Wikipedia API:**
- **Simple Actions:** A limited API is designed with three actions: search[entity], lookup[string] (simulating Ctrl+F), and finish[answer].
- **Intentional Weakness:** This API is **deliberately less powerful** than state-of-the-art retrievers. This design choice **compels the LLM to engage in explicit linguistic reasoning (Thought)** to strategically decide what information to search for and how to navigate within pages, mimicking human interaction with Wikipedia.

Prompting Method

Few-Shot :
- **6 (for HotPotQA) and 3 (for FEVER) randomly selected training cases** were manually converted into **ReAct-format trajectories**.
- These human-crafted trajectories served as **few-shot exemplars** within the prompt.

Trajectory Structure ("Dense Thought"):
- This "dense thought" approach ensures that **reasoning steps are explicitly captured at each stage of problem-solving.**
- Process:
  Based on user input => Within the ReAct framework, it means that the LLM , during its "Thought" step , it generates n utilizes these types of "thoughts to guide its problem-solving process"

- **Question Decomposition:** Breaking down complex queries into sub-goals (e.g., "I need to search x, find y, then find z").
- **Information Extraction:** Identifying relevant details from Wikipedia observations (e.g., "x was started in 1844", "The paragraph does not tell x").
- **Commonsense/Arithmetic Reasoning:** Applying general knowledge or calculations to process information (e.g., "x is not y, so z must instead be...", "1844 < 1989").
- **Search Reformulation:** Adapting search strategies when initial attempts are insufficient (e.g., "maybe I can search/look up x instead").
- **Answer Synthesis:** Combining gathered information to formulate the final response (e.g., "...so the answer is x").

Experimental Cases

- **(a) Standard Prompting:** All ReAct components (Thought, Action, Observation) were removed. This scenario measures the performance of the most basic LLM prompting.
- **(b) Chain-of-Thought (CoT) Prompting:** Action and Observation components were removed from ReAct. This scenario emphasizes the role of 'Thought' (internal reasoning) only.
- **(c) Acting-only (Act) Prompting:** The Thought component was removed from ReAct. This scenario highlights the role of 'Action' (external tool use) without explicit internal reasoning.

Prompt Method[a]	HotpotQA (EM)	Fever (Acc)
Standard	28.7	57.1
CoT (Wei et al., 2022)	29.4	56.3
CoT-SC (Wang et al., 2022a)	33.4	60.4
Act	25.7	58.9
ReAct	27.4	60.9
CoT-SC → ReAct	34.2	**64.6**
ReAct → CoT-SC	**35.1**	62.0
Supervised SoTA[b]	67.5	89.5

Table 1: PaLM-540B prompting results on HotpotQA and Fever.

[a] HotpotQA EM is 27.1, 28.9, 33.8 for Standard, CoT, CoT-SC in Wang et al. (2022b).
[b] (Zhu et al., 2021; Lewis et al., 2020)

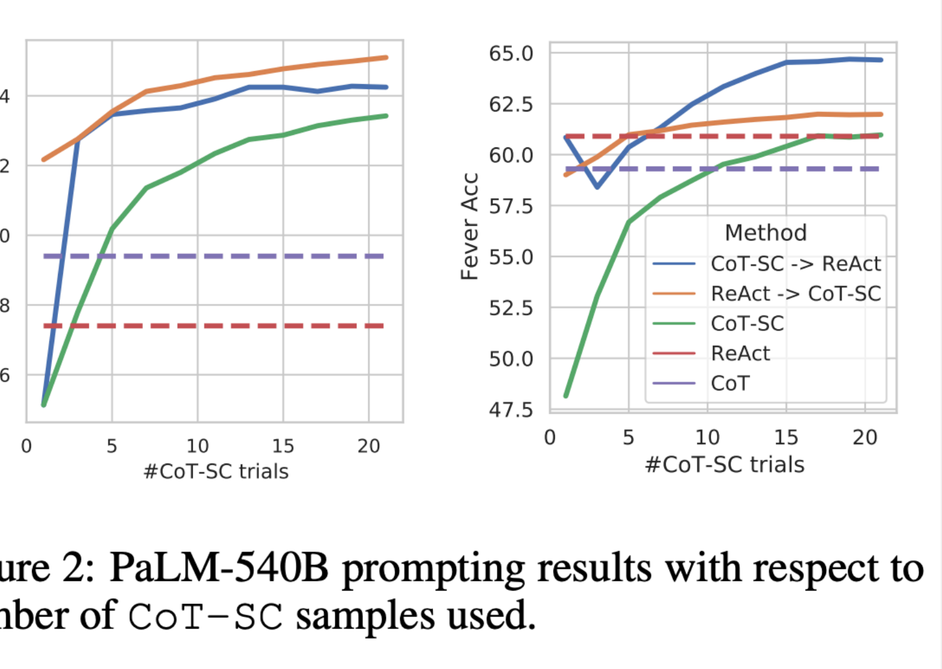

Figure 2: PaLM-540B prompting results with respect to number of CoT-SC samples used.

Combining Internal and External Knowledge => **Core Strengths & Weaknesses Identified**

- **A) ReAct → CoT-SC (Switch from ReAct to CoT-SC):**
- **Condition:** When ReAct fails to return an answer within a predefined number of steps.
- **Action:** The system falls back to using CoT-SC to attempt problem-solving.
- **Step Thresholds:** Set at 7 steps for HotpotQA and 5 steps for FEVER, as further steps did not improve ReAct's performance.
- **Rationale:** If ReAct struggles to find an answer using external search (indicating limitations in external knowledge utilization), the system leverages CoT-SC's strong internal reasoning capabilities to try and solve the problem.
- **B) CoT-SC → ReAct (Switch from CoT-SC to ReAct):**
- **Condition:** When the majority answer among 'n' CoT-SC samples occurs less than 'n/2' times (i.e., the model's internal knowledge does not confidently support the task).
- **Action:** The system falls back to using ReAct to attempt problem-solving.
- **Rationale:** If CoT-SC, despite multiple internal reasoning attempts, cannot converge on a confident answer (indicating uncertainty from internal knowledge alone), the system switches to ReAct to verify facts and augment information using external search, aiming for a more reliable solution.

FineTuning => used **Bootstrapping Approach**

- Used **3,000 trajectories** (sequences of thoughts, actions, observations) that included correct answers.
- These trajectories were **generated by ReAct** (and other baselines) using larger, more capable models.
- This automatically generated data was then used to **finetune smaller language models** (specifically, PaLM-8/62B).
- **Objective of Finetuning:** To enable these smaller models to **decode (generate) full ReAct-like trajectories** (all thoughts, actions, and observations) in response to input questions or claims.
- **Benefit:** Allows the advantages of ReAct to be transferred to more efficient, smaller models without requiring costly manual data annotation.

ReAct Vs CoT
- ReAct **outperforms CoT on FEVER** (60.9 vs. 56.3).
- ReAct **slightly lags CoT on HotpotQA** (27.4 vs. 29.4).
- **CoT's Hallucination issue**
  - Serious issue :
  - positive rate (14%) compared to ReAct (6%) in successful outputs.It constitutes CoT's **major failure mode(56%)**.

	Type	Definition	ReAct	CoT
Success	True positive	Correct reasoning trace and facts	94%	86%
	False positive	Hallucinated reasoning trace or facts	6%	14%
Failure	Reasoning error	Wrong reasoning trace (including failing to recover from repetitive steps)	47%	16%
	Search result error	Search return empty or does not contain useful information	23%	-
	Hallucination	Hallucinated reasoning trace or facts	0%	56%
	Label ambiguity	Right prediction but did not match the label precisely	29%	28%

- **Non-informative search** accounts for 23% of ReAct's error cases, derailing its reasoning and making recovery difficult.
- This highlights a potential **trade-off between factuality (ReAct) and flexibility (CoT)**, which motivated the proposed hybrid strategies.

Hallucinations is a serious problem for CoT

Decision Making Tasks

ALFWorld

ALFWorld serves as a challenging benchmark for evaluating an AI's capacity for complex reasoning, planning, and acting in a dynamic, text-only environment by leveraging commonsense knowledge.

ALFWorld

ALFWorld is a synthetic, text-based virtual environment game designed to test an AI agent's ability to perform complex, multi-step tasks within a simulated household.
Key Characteristics:
- **Text-Based Interface:** All interactions, environment descriptions, and actions are communicated purely through text commands. There are no visual elements.
- **Simulated Household:** Features common rooms (kitchen, living room, bedroom) and objects found in a typical home.
- **Goal-Oriented:** Agents are given high-level objectives (e.g., "examine paper under desklamp," "make coffee with the coffee machine").
- **Complex Interactions:** Achieving goals requires a sequence of precise text commands, such as navigating (go to table), manipulating objects (take paper), and using tools (use coffee machine).
- **Commonsense Reasoning:** Success often relies on applying everyday knowledge (e.g., "books are usually on bookshelves," "refrigerators store food") to infer object locations or appropriate actions.
- **Long-Horizon Planning & Sparse Rewards:** Tasks typically involve many steps, requiring long-term planning. Rewards are "sparse," meaning the agent only receives positive feedback upon successful completion of the entire goal, not for intermediate steps.

Selected ALFWorld to perform well in complex, Langague-based decision making environments

How ALFWorld utlized :
- **Test ReAct's Reasoning and Acting Capabilities:**
- Used as a complex, multi-step environment to evaluate ReAct's ability to plan and execute long action sequences, going beyond simple Q&A to actual decision-making.
- Ideal for demonstrating how ReAct's iterative Thought-Action-Observation cycle functions effectively.
- **Validate Commonsense Knowledge Utilization:**
- Leveraged ALFWorld's reliance on everyday commonsense (e.g., where items are typically found) to show how ReAct, through its Thought component, effectively utilizes the LLM's pre-trained knowledge for efficient action planning.
- **Demonstrate the Importance of "Thoughts":**
- ReAct was prompted with trajectories containing "sparse thoughts" (goal decomposition, subgoal tracking, commonsense reasoning).
- Crucially, it was compared against an "Act" baseline (same trajectories but *without* thoughts).
- This comparison aimed to empirically prove that explicitly incorporating reasoning ("thoughts") significantly enhances performance in complex environments like ALFWorld, quantifying the critical contribution of internal deliberation.
- **Assess Long-Term Planning and Exploration:**
  - ALFWorld's multi-step, long-horizon tasks with sparse rewards were used to evaluate ReAct's capacity for strategic planning, subgoal tracking, and systematic exploration, facilitated by its Thought process.

WebShop

WebShop is a simulated online shopping environment designed to test how well AI can shop like a human.
Key Points:
- It uses **real product data (1.18M items)** and **real human shopping instructions**.
- AI agents must **understand user requests** (e.g., "find a nightstand with drawers, nickel finish, under $140").
- They then **interact with a simulated website** (search, click buttons, select options, buy) to find the right product.
- Unlike simpler environments, WebShop features **noisy, real-world text** (product descriptions, reviews) and requires **complex reasoning** to navigate.
- Its goal is to evaluate AI's ability to **understand complex language, make multi-step decisions, and perform practical tasks** in a realistic web environment.

1 **Benchmarking ReAct's Generalization:**
- It serves as a more challenging environment than ALFWorld (which is cleaner and more structured) to see if ReAct's Thought-Action mechanism can generalize to the complexities of real web interfaces and unstructured text.

2 **Testing Robustness to Real-World Noise and Variety:**
- WebShop's vast amount of real product data, diverse text types (titles, descriptions, user reviews), and complex user instructions create a "noisy" environment. The paper uses this to test ReAct's robustness in extracting relevant information and making decisions amidst this complexity.

4 **Evaluating Complex Instruction Following and Information Extraction:**
- utilizes WebShop's detailed human instructions (e.g., "I am looking for a nightstand with drawers.

Findings : ALFWorld

- **ReAct's Dominant Performance:** ReAct significantly outperformed both Act (no thoughts) and BUTLER (imitation learning), achieving a much higher success rate (ReAct 71% vs. Act 45% vs. BUTLER 37%).
- **Crucial Role of 'Thoughts':** Even the *worst* ReAct performance surpassed the *best* Act performance, clearly indicating that the presence or absence of 'thoughts' has an **absolute, decisive impact** on performance.
- **Consistent Performance Gain:** ReAct showed an **average 62% relative performance gain** over Act, proving that 'thoughts' are essential for goal decomposition and state tracking in complex environments.
- **Conclusion:** ALFWorld unequivocally demonstrated that **ReAct's 'thought'-based reasoning leads to superior performance in complex simulated environments and is critical for task success.**

Findings: WebShop

- **Proof of ReAct's Practical Value:** WebShop confirmed that **ReAct is effective even in noisy, real-world web environments.**
- **Performance Leap with 'Thoughts':** While Act already performed comparably to existing IL/IL+RL methods, ReAct, with its **"sparse reasoning," achieved a significant absolute 10% improvement** in success rate.
- **Evolved Role of 'Thoughts':** In WebShop, ReAct's 'thoughts' were crucial for **identifying relevant information amidst noise and bridging the gap between user instructions and actual product/option details** (e.g., reasoning how to search and filter for specific attributes).
- **Gap with Human Performance:** Despite its improvements, **ReAct still falls short of expert human performance.** Humans perform more extensive product exploration and query reformulation, which remains challenging for current prompting-based methods.
- **Conclusion:** WebShop showed that **ReAct significantly enhances information identification and action decision-making in complex, noisy real-world web environments through 'thoughts,' but it still has limitations in flexible exploration compared to human capabilities.**