# Non-Coding Component

## A. Conceptual Map

Large Language Models (LLMs) are improving and learning how to use external tools to solve problems better. Different methods help LLMs think, plan, and act more effectively. These methods include:

### ReAct (Reasoning + Acting)

This method follows a structured cycle where the model first reasons about a task, then takes an action, observes the outcome, and refines its reasoning based on the result. This continuous reasoning-action loop ensures that the model improves with each iteration. It first generates thoughts, performs an action, observes the response, and updates its reasoning in a structured loop.

### Toolformer

Instead of being explicitly programmed, the LLM learns when and how to use external tools by observing usage patterns in training data. The process involves detecting when an API or tool is useful, invoking it automatically through an API call, and integrating the result back into its response. This structured API-based tool invocation ensures seamless and autonomous tool use.

### Chain of Tools

This method structures tool interactions into a sequence, where the output of one tool serves as the input for another. The model first determines the required tools, then sequentially executes them in an optimized order, ensuring smooth multi-step task execution. This approach enhances efficiency in complex workflows.

### ReST Meets ReAct

This builds upon ReAct by introducing reinforcement learning, allowing the model to improve over time. The process follows an iterative loop where the LLM first reasons, takes action, receives feedback (either AI-generated or external), and then refines its approach based on past experiences. This continuous learning cycle leads to better decision-making.

### LATS (Language Agent Tree Search)

This method enhances planning by exploring multiple potential actions before selecting the best one. The model first generates a tree of possible actions, evaluates their outcomes using Monte Carlo Tree Search, and then picks the most optimal path. This structured flow is beneficial for decision-heavy tasks requiring strategic planning.

## B. Analysis

### ReAct vs. Toolformer

- ReAct allows the model to think before acting, improving decision-making through an iterative reasoning-action cycle. However, it depends on external tools and might not always make the best decisions.

- Toolformer, on the other hand, helps LLMs learn tool usage on their own by invoking APIs at the right time, but it might struggle with new or unseen tools.

### ReST Meets ReAct vs. LATS

- ReST Meets ReAct focuses on iterative learning and self-improvement. It refines responses over time using reinforcement learning, ensuring that the model continuously learns from its mistakes and adapts accordingly. This makes it ideal for applications requiring gradual improvement over multiple interactions.

- LATS, in contrast, prioritizes structured decision-making. It evaluates multiple possibilities at once before making a choice, reducing the chances of errors in a single step. Instead of learning iteratively like ReST Meets ReAct, it performs extensive upfront planning through tree search algorithms. This makes it more suitable for scenarios where selecting the best course of action from the start is more valuable than adapting over time.

**Comparison of AI Model Structure, Thought Process, and Tool Interaction**

|  |  |  |  |
| --- | --- | --- | --- |
| **Model Name** | **Structure & Design** | **Thinking Process** | **Tool Interaction** |
| **ReAct** | Alternates between reasoning and taking actions in an iterative loop | Uses step-by-step thinking before making a decision | Calls external tools after analyzing the situation through multiple iterations |
| **Toolformer** | Learns when and how to use tools independently via API calls | Self-supervised learning to improve reasoning | Automatically integrates APIs and external tools based on past experience |
| **ReST Meets ReAct** | Builds on ReAct by adding self-improvement through reinforcement learning | Refines decision-making through continuous feedback loops | Learns better tool usage by evaluating past interactions iteratively |
| **Chain of Tools** | Connects multiple tools in a sequence | Organizes reasoning into structured workflows | Uses multiple APIs together to complete complex tasks efficiently |
| **Language Agent Tree Search** | Uses tree-based exploration for structured planning | Explores multiple options before picking the best path | Selects tools strategically for optimized decision-making through tree search algorithms |

**Real-World Applications**

**Shoppin Application:** These methods can be integrated into a virtual shopping assistant that helps users navigate e-commerce platforms. For example, **ReAct** can help analyze user preferences and suggest products, **Toolformer** can fetch real-time prices and stock availability, **Chain of Tools** can combine multiple services such as price comparison and delivery estimation, and **LATS** can optimize shopping decisions based on seasonal trends and discounts.

Each of these approaches improves how AI models work, making them more helpful in solving real-world problems.

## C. Open Questions

-**Scalability** – How can we ensure that LLMs handle millions of users at the same time?  
**-Adaptability** – Can these models learn to use new tools without human help?  
-**Error Handling** – How should LLMs respond when a tool gives incorrect or incomplete data?  
-**Integration** – What’s the best way to combine these different approaches into one powerful AI system?

**Possible Improvements**

* Adding **self-correction mechanisms** so the model can recognize and fix its own mistakes.
* Using **smarter decision-making algorithms** to make better choices faster.
* Allowing **human feedback loops** so models can keep improving with real-world data.

By solving these issues, we can make LLMs more powerful and useful in everyday life.