

Large Language Models have transformed our world – can they help to build it?

Graham Long
Thales UK
Crawley, West Sussex
Graham.Long@uk.thalesgroup.com

ABSTRACT

The extraordinarily rapid and disruptive impact of Large Language Models and their ability to generate human-like, conversational responses to text inputs has revolutionized machine interaction with humans. This step change has stimulated tremendous public attention, debate, controversy, and applied innovation of this artificial intelligence (AI) technology to various industries and purposes, from customer service and language translation to creative writing, content creation, and software development.

The state-of-the-art transformer architectures behind large language models are pushing the boundaries of generative AI, providing new ways of directing AI model behaviour. For example, when combined with diffusion models, they deliver improved methods of creating high-quality content, like synthetic images or 3D models, directly from text input. Generative AI's ability to create data and content as plausible, synthetic alternatives to real data offers significant flexibility, lower cost, and faster generation of content, data, and synthetic environments. However, although the capability to create specific AI-generated content may exist, it is the opportunity to incorporate these AI functions into an integrated pipeline that, when combined with AI-based autonomy, has the potential to deliver transformative improvements in efficiency, productivity, and capability.

Current SE pipelines typically employ AI content and data creation processes as independent operations. This paper will build on ongoing research to explore how disruptive AI technologies, like transformer and diffusion architectures together with large language models, may be leveraged and combined with other generative and AI techniques – not only to generate data and content but to provide new, innovative AI-based approaches to directing, integrating and automating these processes. Furthermore, it will consider how these capabilities can be composed into a solution architecture that provides an end-to-end AI-based SE construction pipeline to facilitate the dynamic, faster delivery, and economical production of complex SEs and ultimately enhance their exploitation, user readiness and decision-making.

ABOUT THE AUTHORS

Graham Long has worked in the training and simulation industry for over 30 years. A synthetic environment specialist, he has extensive experience leading the development of visual system solutions, synthetic environments, production tools, processes, and interoperability standards, delivering numerous civil and military simulation programmes and research projects - including the exploitation and utilization of AI within these systems and the convergence and application of simulation and synthetic environment technologies with Digital Twins.

Large Language Models have transformed our world – can they help to build it?

Graham Long
Thales UK
Crawley, West Sussex
Graham.Long@uk.thalesgroup.com

INTRODUCTION

The remarkable unremitting advancements of artificial intelligence (AI) have stimulated significant public attention, speculation, and debate concerning the evolutionary trajectory of the technology, its evolving capabilities, and potentially transformative socio-economic implications. Most of the high-profile progress has been in the field of generative AI, which refers to the capacity of AI to learn the underlying patterns and structures of real data and apply this to the generation of plausible but entirely synthetic versions of this data. Until recently, public awareness and attention focused on generative AI's ability to create highly realistic fake images and videos of people, synonymous with the emergence of "deepfakes." However, generative AI has rapidly expanded its application to many domains and fields - from drug discovery to predicting stock market trends or generating software code. The last two years have seen a further step change in generative AI with the emergence of large language models (LLM). These have exploded into the public domain and suddenly introduced an entirely new medium of human-machine interaction and, perhaps for the first time, exposed tangible AI capability to everyone, democratizing access to the technology and opening many new possibilities for its application and exploitation. Indeed, keeping pace with the rapid velocity of LLM evolution and their applications is challenging.

Nevertheless, there is significant potential to apply these technologies to synthetic environments, particularly to synthetic environment data and construction pipelines. Many suitable generative AI capabilities can and are being leveraged and applied to specific tasks or problems in the SE pipeline. For example, generative computer vision capabilities currently create synthetic versions of real data, improve data resolution, extract cultural feature data from imagery, perform object detection, detect land classes and materials, or generate 3D objects.

Incorporating these valuable capabilities into the SE production process has the potential to greatly enhance the quality, fidelity, and utility of SEs. But what if it was possible to autonomously control and coordinate the execution of these capabilities within a production pipeline, with the flexibility to adapt the pipeline and tasks to variations in input data, SE requirements, or other conditions? - it would initiate a powerful transformation of SEs and SE content generation, improving efficiency, speed, and capability and radically alter development methods. The concept of autonomous agents aligns with the evolution of artificial general intelligence (AGI). Research has exposed the emergent reasoning capabilities of LLMs and raised the possibility of applying LLMs to AGI tasks and exploring them as a candidate technology for autonomous agents.

LARGE LANGUAGE MODELS

Large language models (LLMs) are trained to understand and generate text resembling the form and content of human communication and have revolutionized natural language processing (NLP) tasks. Trained on a massive corpus of text data, they learn to predict and determine patterns in human language. LLMs have many practical applications, their text generation, question-answering and creative writing capabilities now form the foundation of virtual assistants, translation services, and numerous content-generation tools. Notably, they have revolutionized machine-human interaction, providing an entirely new mode of interface that has abruptly made AI accessible to all.

LLM Technology

LLMs are typically based on a transformer architecture. Since their introduction in 2017 (Vaswani et al., 2017), transformers have become recognized as one of the most advanced AI models ever developed and are the enabling technology behind disruptive LLM implementations like ChatGPT.

Transformer architectures aim to solve complex sequence-to-sequence tasks like machine translation, text summarization, and speech recognition. They can develop an intricate comprehension of syntax, semantics, and context by performing sequential data analysis using attention or self-attention techniques to weigh the importance of different words or parts of a sentence in the input text. Furthermore, by simultaneously processing the entire input sequence, transformers can exploit parallel computation and achieve faster training times. Their self-supervised learning approach is one of their key attributes. By learning from raw, unlabelled training data transformers gain access to trillions of images and petabytes of text data available on the web as a training data set – and free themselves from the need for difficult to obtain labelled data.

Transformers have proven to be the state-of-the-art (SOTA) for NLP tasks. However, they are fast becoming AI's "Swiss army knife" as transformer-based sequence modelling approaches expand their application to tabular data, protein sequences, organic molecules, reinforcement learning, and computer vision.

Multi-modal Large Language Models

The emergence of Large language models (LLMs), such as GPT-3, GPT-4, PaLM, LLaMa, and ChatGPT, have demonstrated extraordinary capability and performance in language understanding, generation, interaction, and reasoning. Nevertheless, although this has revolutionized the natural language processing (NLP) field, most current LLMs can only analyze and generate text and need more ability to understand other data types. A lack of multi-modal generative capability (text, images, audio, video, and other sensory input forms) limits the application and utility of LLMs to domains such as SE that have an unavoidable need to consume, process, and generate many modes of data.

However, developing multi-modal LLMs is challenging. They require diverse, high-quality multi-modal training data, and processing this data increases architectural complexity, leading to computationally intensive and time-consuming training. Nonetheless, since September 2022, the DALLE-2, Stable Diffusion, and Imagen, multi-modal LLM text-to-image generators have been publicly released. These use transformer architectures in their diffusion models to deliver SOTA, high-quality synthetic image generation capability. By conditioning the image generation process on text inputs that describe the desired image, they can generate an image that matches the given text description. More recently text-to-3d generators, such as Point-E, have applied these same principles to create 3D models from text.

Foundation Models

DALLE-2, Stable Diffusion, Imagen, GPT-3, GPT-4, PaLM, LLaMa, and ChatGPT represent the next evolution of AI. This new class of "foundation models" will replace the task-specific models that have dominated AI to date. Trained on vast amounts of broad, unlabeled data, this new model class is adaptable to a wide range of downstream tasks with minimal fine-tuning. It can form the foundation for many different applications of the AI model.

AI AGENTS

Advances in multi-modal language models provide very capable solutions to specific tasks, such as text-to-image generation. By specializing in a particular domain and supporting relevant modalities, AI models can acquire expertise and provide more accurate and tailored responses, enabling them to assist users in specific areas of interest effectively. But in reality, many activities and problems are complex tasks composed of many sub-tasks that span many modalities.

For AI to take on the role of managing these complex tasks, such as those that make up an SE production pipeline – it must possess the capability to act with a degree of autonomy, decompose complex goals into sub-tasks, complete them one by one, and use reasoning to interpret perceptions, problem-solve, draw inferences, and determine actions and their outcomes.

Their outstanding ability in language understanding, generation, interaction, and reasoning has created the possibility of using LLMs as the underlying “thinking” engine or brain of an AI agent that can manage complex tasks. But enhanced reasoning capabilities are insufficient on their own. Despite their incredible power, LLMs can struggle with some of the simplest tasks that computers typically excel at – such as logic, calculations, and search. They cannot access up-to-date information on recent events, are unaware of the progression of time, and can hallucinate facts.

This situation has given rise to the concept of AI agents that overcome these inherent LLM limitations by employing the LLM as a controller and general-purpose language interface (Hao et al., 2023) to a set of external expert modules or tools that take on the multi-modal expert tasks the LLM cannot. The agent can solve complex problems using a neuro-symbolic architecture (Karpas et al., 2023) that employs the LLM (neural computation) to decide how to 'route' incoming natural language queries to the appropriate external AI expert tools (neural computation) or tools like calculators, search or weather API database (symbolic computation).

Prompting

Prompts are fundamental to human-machine interaction with LLMs; they provide the mechanism for humans to communicate with LLMs using natural language and text to direct the LLM to generate the desired response to some specific query, task, or action. Designing effective prompts is the key to unlocking the inherent potential and capabilities of LLMs and has given rise to prompt engineering – the crafting of clear and specific instructions or queries to elicit the desired responses from the LLM.

More effective prompting is achieved by following some basic design principles - incorporating instructions, context, input data (which may be an image in the case of DALLE-2), an output format, and using specific, precise, and unambiguous language. Effective prompts ensure GPT-4 responds appropriately to human queries or DALLE-2 creates the expected images. However, prompts and specific prompt engineering methods are also central to the emergent capability of LLMs to perform complex reasoning tasks using their language understanding and generation capabilities. LLM complex reasoning is also vital to the functioning of autonomous agents. In essence, agents provide a mechanism to leverage the ability of LLMs to understand and act on very sophisticated initial prompts. The prompt instructs the LLM to break down the process of performing complex tasks into a sequence of steps resolved one at a time.

Prompting strategies are under constant development and evolution, but in-context learning, chain-of-thought prompting, reasoning-acting and reflection techniques have proven very effective at enhancing LLM reasoning.

In-context learning

In context-learning can improve LLM reasoning by training them to solve tasks by including one or several (few-shot prompting (Brown et al., 2023)) examples of the input-output pairs in the prompt as a demonstration of the task, enabling the LLM to understand better and respond appropriately to specific instructions, references, or queries within the conversation.

Chain-of-thought (CoT)

Chain of thought prompting encourages the LLM to explain its reasoning (Wei et al., 2023) and break down complex problems into intermediate steps that are solved individually. CoT provides a prompt containing an example input and an example output that includes a series of reasoning steps that guide the model to the final output. Zero-shot-CoT takes this concept further – it provides no examples with the prompt (zero-shot) but by appending the words "Let's think step by step." to the end of a question, LLMs can generate a chain of thought that answers the question. These CoT explanations of reasoning enable the LLM to extract more accurate answers.

Reasoning-Acting

In-context learning and CoT reasoning allow autonomous agents to plan tasks and use tools to complete complicated tasks, but LLMs also need to learn how to use external tools better and to learn from mistakes. The ReAct (Reasoning-Acting) (Yao et al., 2023) framework aims to provide the LLM with the ability to reason about the actions they can perform when undertaking tasks in which the LLM can perform specific tasks. For example, an LLM may be able to interact with an external module to retrieve information. When asked a question, the LLM could act to retrieve information and then answer the question based on this retrieved data. This approach enables the LLM to break a user

input down into tasks, reason on the outputs, create sub-tasks, and iterate through this process to achieve the desired result.

Reflection

It is challenging for LLM agents to quickly and efficiently learn from trial-and-error. Reflexion (Shinn et al., 2023) is a proposed approach that leverages verbal reinforcement to teach agents to learn from past mistakes. By reflecting on task feedback signals, and maintaining their own reflective text in memory the LLM can be induced to make better decisions.

Agent Workflow

With the ongoing discovery and research of emergent LLM reasoning capabilities, autonomous agents are now able to plan and prioritize actions and achieve goals without human intervention - through the application of language skills to understand and create information, utilization of short and long-term memory to retain essential details, execution of actions with external tools and reflection on the results. The following generalized LLM-based agent process flow summarizes these capabilities (Figure 1) (Zhao, 2023). This process applies to any agent and can form the basis of an SE agent solution:

1. **Setting Objective:** The user will set a high-level objective that the agent will complete. For example, in the case of a synthetic environment agent, the objective may be: “Build a 3D representation of Gatwick airport.”
2. **Breaking Down the Tasks:** The agent will pass the user instructions and queries as input prompts to the LLM (like GPT-4) which then breaks down the objective into a list of potential tasks to do. This task list may be stored in long term memory, such as a vector database.
3. **Prioritization.** Once there is a list of tasks, the agent will use LLMs’ reasoning skills to evaluate and prioritize the tasks to decide which task it should do next.
4. **Execution.** Once a specific model is assigned to a task, the agent executes the task using an external tool, and sends the result of the execution and gathered information back to the LLM, and will be saved in long term memory.
5. **Evaluation & Creating New Tasks.** If necessary, when the agent completes a task, it will use the LLM reasoning to evaluate the remaining tasks and the results of the previous execution, then determine new tasks that need to be done to achieve the final objective.
6. **Repeat.** Steps 2–5 are repeated until the agent thinks that the original objective is achieved or when the user intervenes.

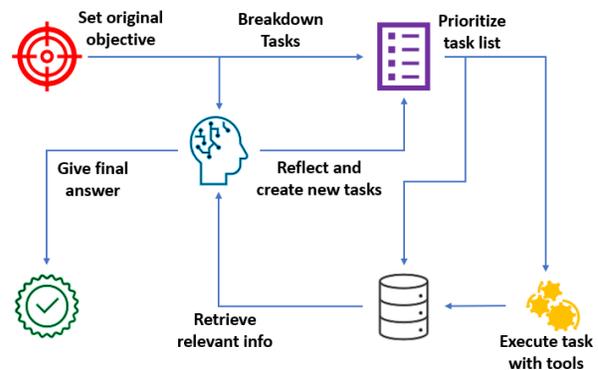


Figure 1 LLM Agent Workflow

APPLICATION TO SE

SE Agent Requirements

An AI agent intended to manage, control, coordinate, and execute a range of processes associated with the data processing, generation, and content creation present within a typical SE generation pipeline will need to satisfy the following high-level requirements. The AI agent must possess the capability to:

- a) Decompose a user objective into tasks and sub-tasks
- b) Identify and establish relationships and dependencies between tasks
- c) Plan and execute tasks in the correct sequence
- d) Decompose, plan and execute single or complex task objectives
- e) Decompose, plan and execute tasks with different scenarios, start points & conditions
- f) Accept multi-modal input – text, images, geo data, video, audio, etc.

- g) Process multi-modal data
- h) Access a set of multi-modal expert AI models and symbolic tools
- i) Interface to expert AI models and symbolic tools
- j) Provide results and feedback to the user
- k) Support Human in the loop interaction

These are demanding requirements, and combining them into a single AI agent solution is a challenging prospect, given the current maturity levels of LLM AI agents. However, many of these SE requirements reflect the principles, capabilities, and attributes common to and needed by most AI agents. Furthermore, at an abstracted level, while the goal of an SE AI agent is to assist the SE generation process, the decomposed lower-level task capabilities use classes and types of expert models commonly employed in many domains.

For example, most agents will use computer vision models to perform common tasks such as object detection, image classification, pixel segmentation, and many generative functions like image-to-image techniques to create synthetic imagery. SE agents will extend these computer vision tasks to incorporate SE-specific capabilities. These may include geoAI expert tools, such as feature extraction, land classification, and SE-focused generative functions to synthesize maps, feature data, point clouds, text-to-image, or super-resolution, and 3D content creation models, such as NeRFs or text-to-3D.

Ensuring that the appropriate set of expert tools is available and accessible to the agent is critical to the agent's ability to complete its goals. While the capabilities provided by AI expert tools will enable the agent to perform many data and content processing and generation tasks successfully, the agent's ability to fulfill the broadest SE-related objectives will require non-AI, symbolic tools to be utilized too – such as search, more SE specific tools like game engines, MatLAB or, as SEs expand to incorporate Digital Twin applications, ingestion of live data feeds from data services and live systems.

The principles and practicalities of utilizing expert tools will have a high degree of overlap and commonality between agents and across domains and are directly transferrable to an SE agent. However, combining and sequencing these expert tools into a correctly coordinated set of tasks and sub-tasks will be much more specific and, in some cases, unique to the goal of generating SE content. Figure 2 illustrates a simplified view of the potential variations and complexity of SE-related tasks, sub-tasks, and sequencing that could be required. Variables such as the input data, available assets, available expert tools, and the goals derived from the SE requirements can create many possible task combinations and pathways that will determine the specific set and order of processes. The agent must handle this complexity and variability to select, plan, execute the correct task set and sequence and generate the expected result.

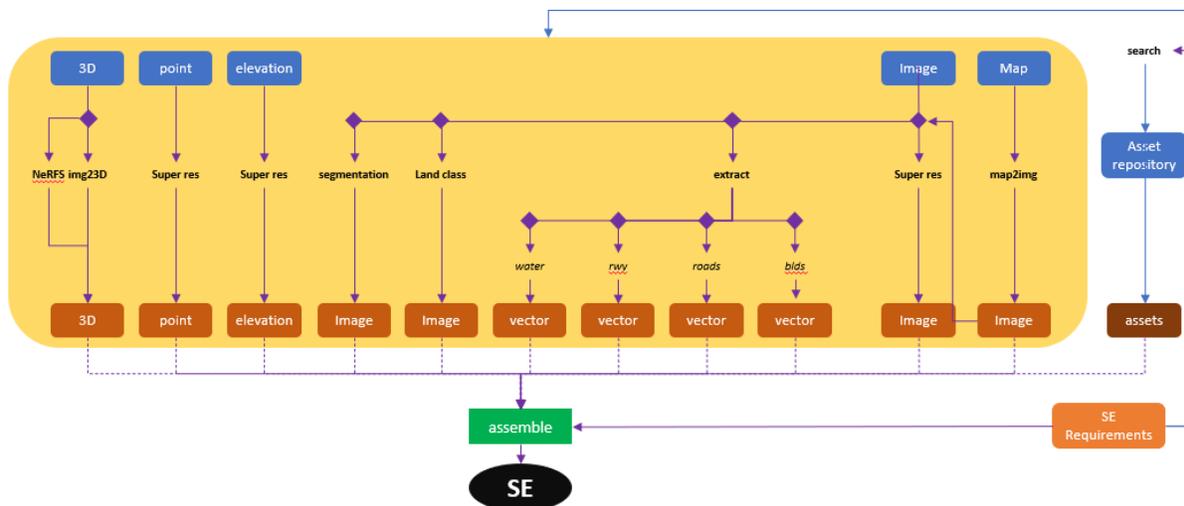


Figure 2 SE Process Flow

SE Agent Architecture

Inspired by the architectures behind recent AI agent research (VisualChatGPT (Wu , et al., 2023), HuggingGPT (Shen et al., 2023) , Toolformer (Schick et al., 2023), and TaskMatrix.AI (Liang et al., 2023)), Figure 3 illustrates a potential SE agent high-level architecture. The concept employs an LLM (such as ChatGPT) to direct user inputs to expert tools and control, manage, and organize their cooperation. The LLM breaks down the user input into a task list. It then allocates expert models to the tasks, initiates their execution, collects results, and responds to the user.

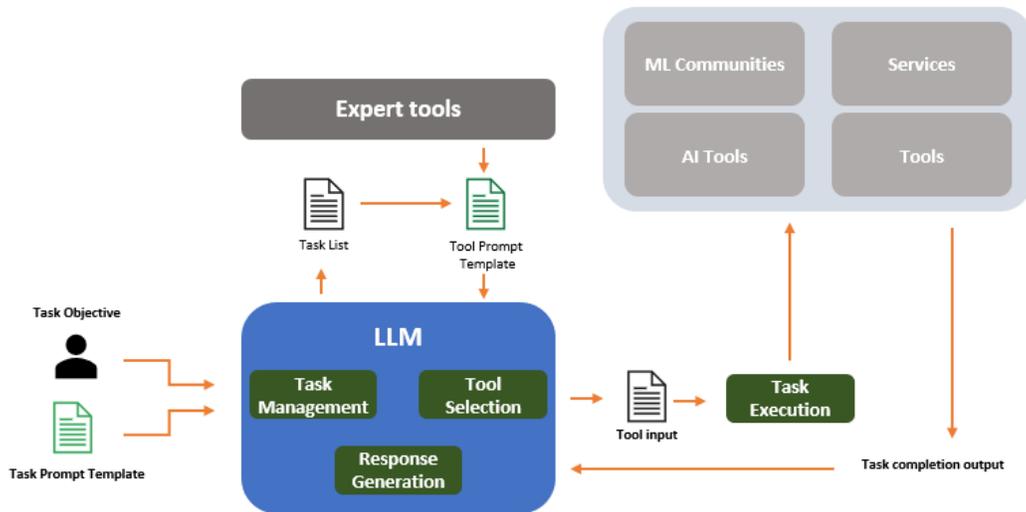


Figure 3 Agent Architecture

The major elements of the architecture are summarized in the following sections. The specific implementations of these stages will vary between agent solutions, but the principles are common.

Task Management

The goal of the first stage of the workflow is to analyze the user request and break it down into a list of structured tasks. A structured prompt template instructs the LLM to parse the user input, identify and select the task from a list of available tasks, and establish other relevant information, such as task resources (images related to the task) and dependencies. Identifying dependencies at this stage enables building the complete task list and connected sub-tasks spanning the whole end-to-end workflow and deploying the coordinated set of model executions. The prompt may also contain some few-shot examples to demonstrate task decomposition into a connected task list with dependencies and resources. The LLM receives the prompt template along with the user input, then processes these, generates the task information, and outputs it in the structured format specified by the prompt.

Figure 4 illustrates a task planning prompt template from HuggingGPT, with the instructional component at the top and the few-shot examples in the demonstrations section below. This prompting approach can be adapted to SE agent applications by fine-tuning the LLM with examples of SE-related user queries and their associated breakdown of tasks, sub-tasks, and dependencies in the context of SEs. The specific details of the LLM task management prompt will vary between agent implementations, depending on the particular prompt engineering approach taken and the design of the structure and details the prompt template must direct the LLM to output.

Prompt	
	#1 Task Planning Stage - The AI assistant performs task parsing on user input, generating a list of tasks with the following format: [{"task": task, "id": task_id, "dep": dependency_task_ids, "args": {"text": text, "image": URL, "audio": URL, "video": URL}}]. The "dep" field denotes the id of the previous task which generates a new resource upon which the current task relies. The tag "<resource>-task_id" represents the generated text, image, audio, or video from the dependency task with the corresponding task_id. The task must be selected from the following options: {{ Available Task List }}. Please note that there exists a logical connections and order between the tasks. In case the user input cannot be parsed, an empty JSON response should be provided. Here are several cases for your reference: {{ Demonstrations }}. To assist with task planning, the chat history is available as {{ Chat Logs }}, where you can trace the user-mentioned resources and incorporate them into the task planning stage.
	Demonstrations
Can you tell me how many objects in e1.jpg?	[{"task": "object-detection", "id": 0, "dep": [-1], "args": {"image": "e1.jpg"}}]
In e2.jpg, what's the animal and what's it doing?	[{"task": "image-to-text", "id": 0, "dep": [-1], "args": {"image": "e2.jpg"}}, {"task": "image-cl", "id": 1, "dep": [-1], "args": {"image": "e2.jpg"}}, {"task": "object-detection", "id": 2, "dep": [-1], "args": {"image": "e2.jpg"}}, {"task": "visual-question-answering", "id": 3, "dep": [-1], "args": {"text": "what's the animal doing?", "image": "e2.jpg"}}]
First generate a HED image of e3.jpg, then based on the HED image and a text "a girl reading a book", create a new image as a response.	[{"task": "pose-detection", "id": 0, "dep": [-1], "args": {"image": "e3.jpg"}}, {"task": "pose-text-to-image", "id": 1, "dep": [0], "args": {"text": "a girl reading a book", "image": "<resource>-0"}}]

Figure 4 Example Task Management Prompt

Expert Tool Selection

Following task identification and planning, the expert tool selection stage selects the most appropriate tool. A tool selection prompt incorporating the task information and the user input instructs the LLM to choose the most suitable tool from an available tool list based on a description - which can be a description of an expert AI model (HuggingGPT, VisualChatGPT) or an expert model API (TaskMatrix.AI).

Task Execution

Once selected, the agent inputs the task argument to the models, executes them, and returns the results to the LLM. This process can also allocate the outputs of predecessors to dependent tasks to ensure coherent and successful completion of the entire process chain.

Response Generation

After completing the task executions, the agent composes a final natural language response related to the original user request that combines all information from the task planning, model selection, and task execution in a structured format appropriate to the task.

Implementation

In practise, there are different approaches to implementing this architecture, but not all are suitable for SE applications. For example, AutoGPT, AgentGPT and BabyAGI are not multi modal and do not provide access to expert AI models. Instead, they interface to what can be considered as “standard” tools, such as search or calculator applications. This would rule their current implementations out of consideration for an SE solution.

The HuggingGPT, VisualchatGPT, and TaskmatrixAI implementations are more relevant to an SE solution but adopt different approaches to the type of tools they can access and the interfaces with them. HuggingGPT and VisualchatGPT only focus on using AI expert models and do not use symbolic tools. VisualChatGPT concentrates entirely on the visual modality and interfacing with visual foundation models. HuggingGPT aspires to utilize any AI expert model but currently supports visual, audio, and text modality models selected from the HuggingFace AI model repository. In contrast, TaskMatrixAI is an evolution from VisualchatGPT, with a more ambitious approach aiming to provide the potential to access any AI or symbolic tool. These differences in the agent's scope and accessible expert tools influence the interface solutions adopted by each agent solution. They all rely on instructing the LLM to create structured text that forms the input to the expert tools. However, HuggingGPT and VisualchatGPT select tools based on a text description of an expert AI model and pass structured task information as input to the chosen model. In contrast, TaskMatrixAI must adopt a universal interface to use any AI or symbolic tool and select an expert tool based on an API description. TaskMatrixAI must retain an API platform to store API descriptions which it uses to instruct the LLM to create the API call as input to the expert tool.

Each of these approaches has its strengths and weaknesses. Limiting access to expert AI models allows the input to the expert models to be kept relatively simple. Providing the model descriptions are available this a flexible and straightforward method of integrating expert models from anywhere - AI communities or private model libraries. However, using an API interface, potentially any expert model is now available for use by the agent. But this does require the generation of more complex inputs and maintenance of an API library for the agent – and requires developers to provide an API to their tool. A HuggingGPT approach provides a suitable short-term solution for an SE agent that only needs access to expert AI models and multi-modal support. However, the TaskMatrixAI API approach offers the most capable long-term solution. With the powerful ability to access and incorporate any tool into the SE agent pipeline, the API approach presents the most flexibility and opportunity for the agent to provide solutions to complex and diverse user requirements.

Examples

The following examples illustrate some of the proposed agent architectures' current capabilities to identify, decompose, control, and coordinate tasks, achieve desired outcomes, access expert tools, and support multiple modalities. The example tasks are not explicitly related to synthetic environments but demonstrate the architectural principles, task types, task planning, modalities, and expert model capabilities the agent can apply to SE tasks. Similar SE domain queries, with SE data and user instructions, can be fulfilled by ensuring access to the required expert tools and adapting the agent through in-context learning to fine-tune the LLM to predict SE workflows and task breakdowns.

Query: In example2.jpg, what's the animal and what's it doing?

Response: Sure, the animal in the image is a tiger and it's laying down. I performed 3 tasks including image-to-text, image-classification, and visual-question-answering on the image you provided. To accomplish this, I selected models because they are the perfect models for this use case respectively. According to the results, I have 88% confidence that the animal is a tiger and also have a 77.8% confidence that it's laying down. The generated image with predicted box is stored at the path: /images/d59a.jpg.



Task Planning:

- 1: [{"task": "image-classification", "id": 0, "dep": [-1], "args": {"image": "example2.jpg"}},
- 2: [{"task": "visual-question-answering", "id": 1, "dep": [-1], "args": {"text": "What is the animal doing?", "image": "example2.jpg"}},
- 3: [{"task": "image-to-text", "id": 2, "dep": [-1], "args": {"image": "example2.jpg"}},

Execution Results:

1. [{"score": 0.8858247399330139, "label": "tiger, Panthera tigris"}, {"score": 0.10940514504909515, "label": "tiger cat"}, {"score": 0.0006216467591002584, "label": "jaguar, panther, Panthera onca, Felis onca"}, {"score": 0.0004262701258994639, "label": "dhole, Cuon alpinus"}, {"score": 0.00030842673731967807, "label": "lion, king of beasts, Panthera leo"}]
2. [{"answer": "sitting", "score": 0.8294351696968079}, {"answer": "laying down", "score": 0.7387925386428833}, {"answer": "resting", "score": 0.625247061252594}, {"answer": "laying", "score": 0.6114301681518555}, {"answer": "relaxing", "score": 0.44740357995033264}]
3. {"generated_text": "a brown and white striped zebra laying on a rock"}

Figure 5 Two Task Example

Figure 5 demonstrates the handling of a computer vision use case with a relatively simple user request containing two tasks. In this case, the HuggingGPT agent reasons that it should break the request into three tasks and organizes and executes three expert models (an image classification, visual-question-answering and image-to-text model). The LLM then aggregates the model inference results and responds to the user with a detailed breakdown of the LLM's reasoning for its task planning and model selection. (The example also details the composition details and structure of the task plan).

The more complex computer vision task in Figure 6 illustrates successfully parsing an abstract user request into tasks with resource dependencies between tasks. In this case, the agent has identified the query requires pose detection, image captioning, and pose conditional image generation tasks. It also correctly determines that task 3 is dependent on the output of task 2 and injects the output image from the parent into the dependent task.

The ability of an agent to receive a sparse user input that may consist of a starting data element and a simple description of the objective, then reason and identify a process to satisfy the goal, is of particular value to an SE context. A possible SE scenario may pass an area map to the agent and instruct it to create a 3D terrain representation of the area - with the expectation that it can determine what it needs to do with the map and the tasks it must undertake to create the terrain representation. Figure 7 demonstrates a sparse input scenario for an out-painting task performed by TaskMatrix.AI. In this case, the user inputs an image and simple instructions to extend it to 2048 x 4096. The agent defines a solution outline consisting of three APIs: Image Question Answering, Image Captioning, and Replace Objects from Image. By iteratively executing the four pre-defined steps in the

Query: based on the pose image of example1.jpg and the description of example2.jpg, please generate a new one.




Response: Based on the pose image of example1.jpg and the description of example2.jpg, I have generated a new image s3f8.jpg for you. The process of generating this image involved using the model to extract the pose information from the basketball image, and then using the model to generate a description of the boy in the image. Finally, I used the model to combine the pose information and the description to generate the final image. The complete path of the generated image is s3f8.jpg. If you have any other questions, please let me know.

Task Planning:

- 1: [{"task": "pose-detection", "id": 0, "dep": [-1], "args": {"image": "example1.jpg"}},
- 2: [{"task": "image-to-text", "id": 1, "dep": [-1], "args": {"image": "example2.jpg"}},
- 3: [{"task": "pose-text-to-image", "id": 2, "dep": [0, 1], "args": {"text": "<resource>-1", "image": "<resource>-0"}},

Execution Results:

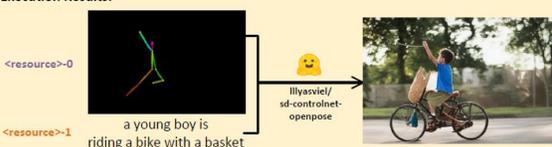


Figure 6 Task & Dependency Example

Human: 

TaskMatrix.AI: Received 35f5f431.png

Human: Extend it to 2048x4096

TaskMatrix.AI: Here is the extend image for you.

Intermediate Thoughts:

Image Question Answering

Image: 

Question: What is the background color of this image?

Answer: Green.

Question: What is the style of the image?

Answer: Flower.

Image Captioning:

Image: 

Caption: a pink flower with green leaves and a purple flower.

LLM Merged Insight:

The painting is a vibrant and colorful scene of two flowers in a lush green garden. The pink flower is in full bloom, its petals a deep rose color and its leaf a bright emerald green. The purple flower in the center is slightly smaller, its petals a deep amethyst and its center a bright yellow.

Replace Objects from Image:

Image: 

Replace Objects: Background

Caption: "LLM Merged Insight"

Output Image: 

Iterative Execution...



Figure 7 Sparse Task Planning

solution outline (the intermediate thoughts column in Figure 7), TaskMatrix.AI can generate a high-resolution image of the desired size.

These examples highlight capabilities that are critical to any SE solution. Firstly, the architecture can support multi-modal capabilities, combining text and visual modalities as input and output. Even though the underlying LLM model (ChatGPT) only supports language, in these cases, both HuggingGPT and TaskMatrix.AI provide extended modalities without retraining the core LLM. Most importantly, the agents demonstrate that they can successfully reason across different scenarios and user query requests to build connected tasks, select suitable but different expert models (five unique expert models chosen for the six tasks), and generate expected results.

Limitations and Benefits

This architecture is heavily reliant on the capabilities of the chosen LLM. It plays a critical role in task planning, the fundamental enabler of the entire approach. Therefore, the LLM must be optimized to generate a feasible plan, which places high importance and demand on prompt engineering. Executing the workflow through the LLM also results in multiple interactions during task execution, increasing the time cost for response generation. Furthermore, LLMs limit the maximum token length (the basic units of text/code for LLMs to process and generate language); this creates a need to be frugal with items like model descriptions and to find ways to summarize such content effectively. Finally, LLMs are capable but imperfect and can provide incorrect predictions or fail to follow instructions which may compromise the task and process workflow.

Nevertheless, the proposed use of an LLM at the core of the agent architecture, with external AI models providing expert AI functionality, leverages the benefits of the foundation model concept. Utilizing their pre-trained capabilities avoids the demanding training of new models or re-training existing models. Furthermore, the domain knowledge of the LLM can be enhanced and tailored toward SE requirements by exploiting its in-context learning ability, which will improve the performance of task planning and execution. The solution does place a dependency on effective prompt engineering. However, the introduction of frameworks, such as Langchain, Dust, Steamship, and more, is intended to simplify the development of applications using LLMs by creating an underlying orchestration engine for LLM interactions accessed via an API.

CONCLUSION

The rapid progress of generative AI in general and the swift emergence and impact of large language models, in particular, has created significant opportunities to transform human-machine interaction and exploit and apply this technology in novel and innovative ways. Applying autonomous AI agents to complex tasks and processes such as SE generation offers transformative benefits regarding efficiency, productivity, and capability improvements. The emergent capabilities of LLMs have revealed them as one potential enabler of AI agent solutions. In less than two years, using language as the interface to connect LLMs with expert AI models and expert symbolic tools has stimulated the development of LLM-based agents. This paper has outlined a potential LLM agent architecture for SE generation activities. Inspired by recent research and experimental solutions, it adopts the principle of an LLM controller that connects with external expert tools to overcome the inherent limitations of LLMs and utilize expert models to perform specific tasks, such as computer vision operations. It exploits emergent LLM reasoning capabilities to plan and decompose complex tasks derived from the user input. Although immature and experimental, this agent concept already demonstrates enormous potential in solving tasks and providing novel, autonomous capabilities. By selecting and accessing suitable AI and symbolic expert tools and fine-tuning the LLM with in-context learning of the SE domain, this architecture is transferrable to SE tasks and processes. Further research on the potential to apply this to SE workflows is required, particularly to assess the achievable level of task complexity and sophistication. However, given its rapid progress and capability, this architecture promises to pave a new course toward autonomous SE processes.

REFERENCES

- Brown T, Mann B, Ryder N, Melanie Subbiah M. (2020). Language Models are Few-Shot Learners. Retrieved from: <https://arxiv.org/abs/2005.14165v4>
- Hao Y, Song H, Dong L, Huang S, Chi Z, Wang W, Ma S, Wie F. (2022) Language Models are General-Purpose Interfaces. Retrieved from: <https://arxiv.org/abs/2206.06336v1>
- Karpas E, Abend O, Belinkov Y, Lenz B, Lieber O, Ratner N, Shoham Y, Bata H, Levine Y, Leyton-Brown K, Muhlgaay D, Noam, Schwartz R, Gal, Shai S, Shwartz S, Amnon Shashua A, Tenenholz M. (2022). MRKL Systems. Retrieved from: <https://arxiv.org/abs/2205.00445v1>
- Liang Y, Wu C, Song T, Wu W, Xia Y, Liu Y, Ou Y, Lu S, Ji L, Mao S, Wang Y, Shou L, Gong M, Duan N. (2023). TaskMatrix.AI: Completing Tasks by Connecting Foundation Models with Millions of APIs. Retrieved from: <https://arxiv.org/abs/2303.16434v1>
- Schick T, Dwivedi-Yu J, Dessì R, Raileanu R, Lomeli M, Zettlemoyer, L Cancedda N, Scialom T. (2023). Toolformer: Language Models Can Teach Themselves to Use Tools. Retrieved from: <https://arxiv.org/abs/2302.04761v1>
- Shen Y, Song K, Tan X, Li D, Lu W, Zhuang Y. (2023). HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face. Retrieved from: <https://arxiv.org/abs/2303.17580v3>
- Shinn N, Cassano F, Labash B, Gopinath A, Narasimhan K, Yao S. (2023). Reflexion: Language Agents with Verbal Reinforcement Learning. Retrieved from: <https://arxiv.org/abs/2303.11366v3>
- Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L., Gomez A, Kaiser L, Polosukhin I. (2017). Attention Is All You Need. Retrieved from: <https://arxiv.org/abs/1706.03762>
- Wei J, Wang X, Schuurmans D, Bosma M, Ichter B, Xia F, Chi E, Le Q, Zhou D. (2023) Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. Retrieved from: <https://arxiv.org/abs/2201.11903v6>
- Wu C, Yin S, Qi W, Wang X, Tang Z, Duan N. (2023). Visual ChatGPT: Talking, Drawing and Editing with Visual Foundation Models. Retrieved from: <https://arxiv.org/abs/2303.04671v1>
- Yang Z, Li L, Wang J, Lin K, Azarnasab E, Ahmed F, Liu Z, Liu C, Zeng M, Wang L. (2023). MM-REACT : Prompting ChatGPT for Multimodal Reasoning and Action. Retrieved from: <https://arxiv.org/abs/2303.11381v1>
- Yao S, Zhao J, Yu D, Du N, Shafran I, Narasimhan K, Cao Y. (2023). REACT: Synergizing Reasoning And Acting In Language Models. Retrieved from: <https://arxiv.org/abs/2210.03629v3>
- Zhao G (2023). A comprehensive and hands-on guide to autonomous agents with GPT. Retrieved from: <https://bootcamp.uxdesign.cc/a-comprehensive-and-hands-on-guide-to-autonomous-agents-with-gpt-b58d54724d50>