International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 11, pp 603-610, November-2023

https://doi.org/10.47001/IRJIET/2023.711080

Smart Automation Using LLM

[1]Priya Ethape, [2]Riya Kane, [3]Ghanashyam Gadekar, [4]Sahil Chimane

[1,2,3,4]Smt. Indira Gandhi College of Engineering, Navi Mumbai, Maharashtra, India

***Abstract* - Smart Automation Using Large language models (LLMs) have emerged as a transformation AI technology, achieving state-of-the-art performance across natural language processing tasks. This survey summarizes the development progress on the General Computer Automation Using Large Language Models project. It aims to create an intelligent agent for automating computer tasks by leveraging large language models. The modular architecture includes components for conversational intelligence, document handling, and application control. Open AI's GPT-3 is integrated for natural language capabilities. Trust in AI agents has been extensively studied in the literature, resulting in significant advancements in our understanding of this field. However, the rapid advancements in Large Language Models (LLMs) and the emergence of LLM-based AI agent frameworks pose new challenges and opportunities for further research. In the field of process automation, a new generation of AI-based agents has emerged, enabling the execution of complex tasks.**

Keywords: LLM, AutoGPT, General computer autonomous.

1. Introduction

Automation has become an integral part of our lives, simplifying various tasks and increasing efficiency. One emerging trend in automation is the use of speech to control desktop computers. This survey report explores the concept of using chatbots as an interface to automate tasks on desktop computers. Artificial intelligence and automation, and it tries to demonstrate to the audience how both Artificial intelligence and automation are related and how they can be more effective when they work together and can give competitive advantage. Artificial Intelligence (AI) is the science that enables computers and machines to learn, judge and use their own reasons. As technologies are becoming more complex, the demand for Artificial Intelligence is growing because of its ability to solve complex problems with limited human resources and expertise and within a limited time. AI adopts the abilities to equip the technical expertise and can amplify expertise to learned deploy new methods and applications. It uses a method called search and pattern matching where the computer is instructed to search its knowledgebase based on the match found and if specific conditions are met to solve a problem.

In the last couple of years, the field of artificial intelligence (AI) has been advancing a tan exponential rate. Large language models (LLMs) based on generative pre-trained transformer architectures such as OpenAI's GPT-3 [1], and more recently ChatGPT [2]'s and GPT-4 [3] have shown astonishing capabilities in understanding and generating language, meeting or exceeding expert-level skills in content generation, question answering, and even discovering new scientific knowledge. More recently, advanced frameworks have emerged that enable the creation of autonomous AI agents. AutoGPT [4], Agent- GPT[5], LangChain [6], and ChatGPT Plugins [7], are capable of receiving high-level objectives, decomposing them into sub-tasks, and interacting online with applications, services, APIs, and tools.

The key modules implemented so far include the Main Controller, Document Handling, OpenAI Chatbot, and App Control. Current capabilities enable natural conversations, local document search, and launching applications through voice commands. The foundation has been laid to further enhance the automation agent with additional features like voice input/output, graphical user interface, computer vision integration, and expanding the document search scope.

Overall, the core modules required for an automated computer agent with custom functionality have been completed. This project has provided hands-on experience in leveraging large language models for computer task automation. The progress so far represents important strides towards developing an extensible automation solution using cutting-edge AI.

2. About LLM

Large language models (LLM) are very large deep learning models that are pre-trained on vast amounts of data. The underlying transformer is a set of neural networks that consist of an encoder and a decoder with self- attention capabilities. The encoder and decoder extract meanings from a sequence of text and understand the relationships between words and phrases in it.

Transformer LLMs are capable of unsupervised training, although a more precise explanation is that transformers perform self-learning. It is through this process that transformers learn to understand basic grammar, languages, and knowledge.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 11, pp 603-610, November-2023

https://doi.org/10.47001/IRJIET/2023.711080

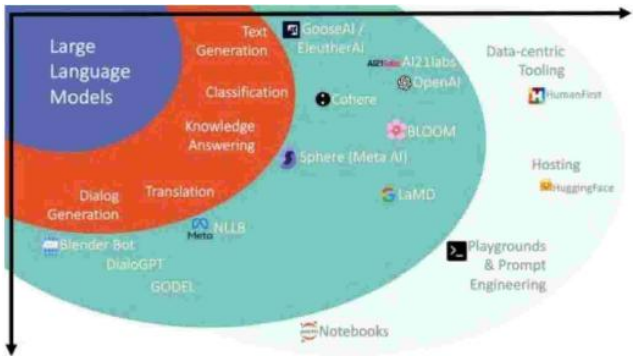

Figure 1: LLM (Large language model)

Unlike earlier recurrent neural networks (RNN) that sequentially process inputs, transformers process entire sequences in parallel. This allows the data scientists to use GPUs for training transformer-based LLMs, significantly reducing the training time.

Transformer neural network architecture allows the use of very large models, often with hundreds of billions of parameters. Such large-scale models can ingest massive amounts of data, often from the internet, but also from sources such as the Common Crawl, which comprises more than 50 billion web pages, and Wikipedia, which has approximately 57 million pages.

Figure 2

LLMs are now capable of comprehending and generating language many like humans, additional considerations of trust that have not been the focus of previous frame works may require attention. We therefore ask, in this quickly evolving landscape of AI-based automation agents, what new challenges and opportunities lie ahead for research? This paper explores these new challenges and opportunities.

RNN: In the context of smart automation for speech control, RNNs can be employed to process and interpret spoken commands. The network's recurrent connections enable it to consider context from previous words or phrases, enhancing its ability to comprehend the nuances of natural language. This makes RNNs well-suited for applications like voice-activated assistants or systems that respond to spoken instructions.

Similarly, in text-based automation, RNNs can be utilized to analyze and generate text in a way that captures the contextual dependencies within the language. This is particularly useful for tasks such as natural language understanding and automated response generation in chatbots.

Dataset: Automation may or may not be based on artificial intelligence. The whole practice of automation has evolved into its current form between the first and third industrial revolution. It involves production using automatic testing, mechanical labour, control systems, computers and operating equipment. All the types of automation which have manifested all around us are bound using explicit programming and rules. To ensure that the same thing becomes an AI, all that needs to be done is to power it up using data. Huge quantities of data, like using neural networks, graphs and deep machine learning must be put in the software. Your coding levels will certainly decide just how much you will be able to make your system stimulate like a human. But most likely, you will be teaching the system all that you already know. In the case of automatic, you will be able to easily know the output using sensor readings. But in case of AI there is always a little bit of uncertainty, just like it's there with the human brain.

Tangibility: LLM agents primarily interact through text or voice, incorporating tangibility through avatars or visual representations enhances the user experience by providing a more relatable and engaging interface. A human-like avatar appearance can facilitate better communication, as users tend to respond more positively to visually recognizable entities. It creates a sense of familiarity and relatability, fostering a stronger connection between the user and the AI agent. Tangibility also helps in conveying non-verbal cues and emotions, enriching the interaction and making it more intuitive. By incorporating avatar-like appearances, LLM-based AI automation agents can create a more immersive and satisfying user experience, improving communication and building trust with users. To achieve tangibility, the following dimensions require special consideration and attention:

Major Components of AI in Automation: An automation system functions using the three components of artificial intelligence. So, depending on the need, they can be either combined or even used separately to allow for a fully automated response. Machine Vision: This refers to the potential of any program to understand what the visual input is. The machine makes use of the training data (images) as a type of foundation for the identification or classification mechanism. Natural Language Processing: Machine language world on the visuals, Natural Language Processing (NLP) does the same to understand human voice and text inputs. It's now possible for machines to understand what the context behind the communication is being carried out and then take actions

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 11, pp 603-610, November-2023

https://doi.org/10.47001/IRJIET/2023.711080

based on the kind of pre-built data and contextual variables which are at play. Some examples of this are Apple's Siri, Amazon Alexa etc.

3. Methodology

Smart agent is a Large Language Model that can be build for handling such task based on computer or web. It will develop voice based searching and task performing on computer. Follow the voice command smart automation API performs general task like web search, app open/close, text generation etc. it'll be able to do general computer tasks Cut the middle man: we don't need a web browser, but rather simply point out that problems like Generation, Search & Agents are getting very simple, and many more will follow. Our media coverage is a testament to our commitment to pushing the boundaries of what's possible with AI and the web. Super impressive, this AI agent can use the web browser like a human. Just describe what you want it to do, and it will automatically operate Chrome for you to achieve your task. Around a decade after virtual assistants like Siri and Alexa burst onto the scene, a new wave of AI helpers with greater autonomy is raising the stakes, powered by the latest version of the technology behind ChatGPT and its rivals. Advances in AI, The new assistants - "agents" - promise to perform more complex personal and work tasks when commanded to by a human, without needing close supervision. Early efforts are only a taste of the sophistication that could come in future years from increasingly advanced and autonomous agents as the industry pushes towards an artificial general intelligence (AGI) that can equal or surpass humans in myriad cognitive tasks.

"The real challenge is building systems with robust reasoning,"

The race towards increasingly autonomous AI agents has been supercharged by the March release of GPT-4 by developer OpenAI, a powerful upgrade of the model behind ChatGPT - the chatbot that became a sensation when released last November.

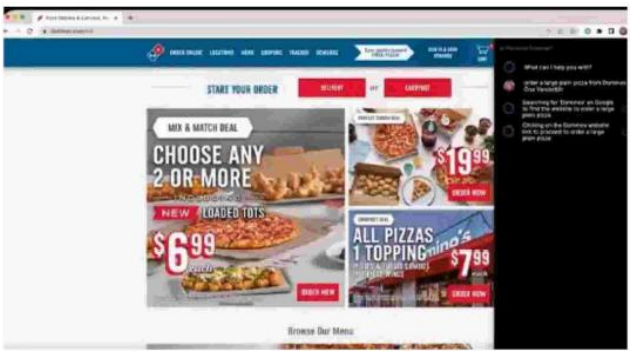

Figure 3

GPT-4 facilitates the type of strategic and adaptable think- ing required to navigate the unpredictable real world. Early demonstrations of agents capable of comparatively complex reasoning came from individual developers who created the BabyAGI and AutoGPT open-source projects in March, which can prioritize and execute tasks such as sales prospecting and ordering pizza based on a pre-defined objective and the results of previous actions. Ever thought ChatGPT could order you dinner? Turns out it can, google search in easy with voice command and even do lots more,

Even if your LLM can do 128K context window, you still want to keep the context as short as possible.

Also, there's been more related research with "lost in the middle", although this is probably because of the way we train instruct models to pay attention to the beginning and the end (where instructions normally are)

While recent language models have the ability to take long contexts as input, relatively little is known about how well they use longer context. We analyze language model performance on two tasks that require identifying relevant information within their input contexts: multi-document question answering and key-value retrieval. We find that performance is often highest when relevant information occurs at the beginning or end of the input context, and significantly degrades when models must access relevant information in the middle of long contexts.

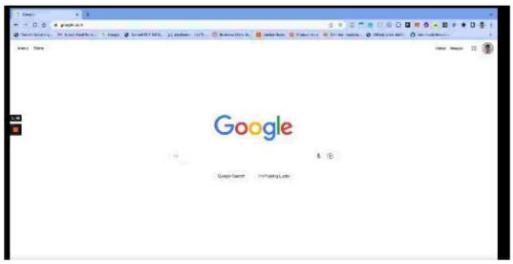

Figure 4

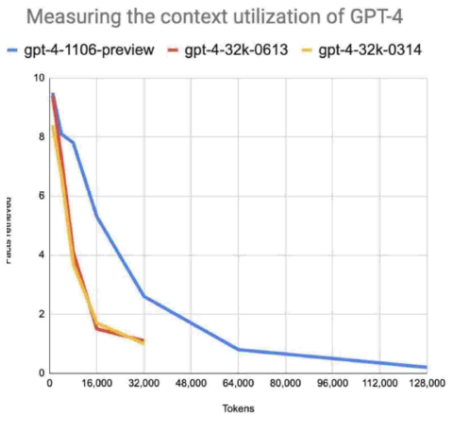

Figure 5

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 7, Issue 11, pp 603-610, November-2023
https://doi.org/10.47001/IRJIET/2023.711080

Furthermore, performance substantially decreases as the input context grows longer, even for explicitly long-context models. Our analysis provides a better understanding of how language models use their input context and provides new evaluation protocols for future long-context models.

Python and JS SDK to build on top of the API and seamlessly integrate with top toolkits such as LangChain and LLamaIndex.

4. Dependencies

Numpy (Use matrix math operations) PyTorch (Build Deep Learning models)
Datasets (Access datasets from huggingface hub) Huggingface_hub (access huggingface data & models) Transformers (Access models from HuggingFace hub)
Trl (Transformer Reinforcement Learning. And fine-tuning.)
Bitsandbytes (makes models smaller, aka 'quantization') Sentencepiece (Byte Pair Encoding scheme aka 'tokenization')
OpenAI (Create synthetic fine-tuning and reward model data)
TVM (Tensor Virtual Machine, converts onnx model to efficient cross-platform use)
Peft(Parameter Efficient Fine Tuning, use low rank adaption (LoRa) to fine-tune)
Onnx (Convert trained model to universal format)

Features:

Effortless Web Automation: Navigate, scrape, and manipulate the web with ease.

AI Agents: Build and deploy custom AI agents for complex tasks.

SDK Support: Native Python and JavaScript SDKs available.

Rich Ecosystem: Seamlessly integrate with LangChain, LlamaIndex, and more.

Global Community: Get support and collaborate with like-minded developers in our Discord.

5. Implementation

Main Controller: Built using Python for rapid development. Uses NLU for intent identification from user queries. Maintains and switches between various operating modes. Routes user input to appropriate module.

Document Handling: Utilizes PDF Miner and Tika libraries for text extraction from documents. Indexes extracted text, maintains local search index. Performs keyword based search to find relevant documents.

Query and solutions model: Uses GPT-3 LLM to integrate conversational capabilities. Stateless design treats each user input as prompt for model. Contextual responses generated with 6000+ token context App Control: Builds index of app info like name and path in a JSON config file. Enables opening apps via voice commands by invoking the stored paths. Provides easy way to add/remove apps by modifying JSON.

Intelligent query solutions: Near human-level conversations using GPT3 LLM. Local Document Search: Search PDFs and text by keyword extraction. External App Launching: Launch apps through voice commands using a JSON con- fig. Mode Switching: Change seamlessly between features like chat, search, launchapps.

Langchain:

Agents involve an LLM making decisions about which Actions to take, taking that Action, seeing an Observation, and repeating that until done. LangChain provides a standard interface for agents, a selection of agents to choose from, and examples of end-to-end agents.

Loading: First we need to load our data. Use the LangChain integration hub to browse the full set of loaders.

Splitting: Text splitters break Documents into splits of specified size

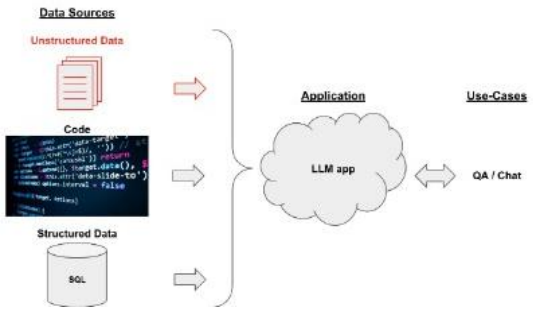

Figure 6

Storage: Storage (e.g., often a vector store) will house and often embed the splits.

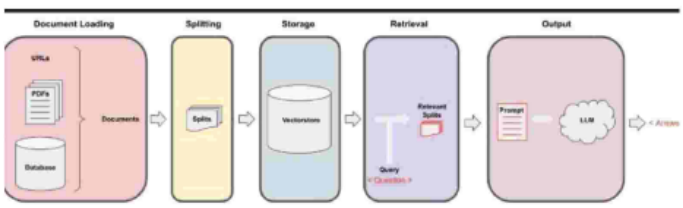

Figure 7

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 11, pp 603-610, November-2023

https://doi.org/10.47001/IRJIET/2023.711080

First set environment variables and install packages:

pip install langchain open ai chromadb langchainhub

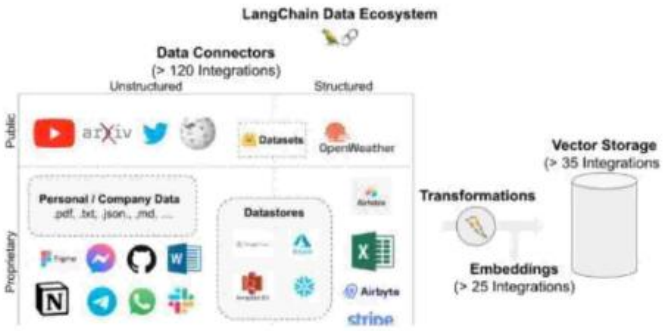

Figure 8

Retrieval: The app retrieves splits from storage (e.g., often with similar embeddings to the input question)

Generation: An LLM produces an answer using a prompt that includes the question and the retrieved data.

LangChain Libraries: The Python and JavaScript libraries. Contains interfaces and integrations for a myriad of components, a basic run time for combining these components into chains and agents, and off-the-shelf implementations of chains and agents. LangChain Templates: A collection of easily deployable reference architectures for a wide variety of tasks. LangServe: A library for deploying LangChain chainsas a RESTAPI.

LangSmith: A developer platform that lets you debug, test, evaluate, and monitor chains built on any LLM framework and seamlessly integrates with LangChain.

Chains go beyond a single LLM call and involve sequences of calls (whether to an LLM or a different utility). LangChain provides a standard interface for chains, lots of integrations with other tools, and end-to-end chains for common applications.

Data Augmented Generation:

Data Augmented Generation involves specific types of chains that first interact with an external data source to fetch data for use in the generation step. Examples include summarization of long pieces of text and question/answering over specific data sources.

Agents:

Agents involve an LLM making decisions about which Actions to take, taking that Action, seeing an Observation, and repeating that until done. LangChain provides a standard interface for agents, a selection of agents to choose from, and examples of end-to-endagents.

Memory:

Memory refers to persisting state between calls of a chain/agent. LangChain provides a standard interface for memory, a collection of memory implementations, and examples of chains/agents that use memory.

Evaluation:

[BETA] Generative models are notoriously hard to evaluate with traditional metrics. One new way of evaluating them is using language models themselves to do the evaluation. LangChain provides some prompts/chains for assisting inthis.

Why is LangChain important?

LangChain is a framework that simplifies the process of creating generative AI application interfaces. Developers working on these types of interfaces use various tools to create advanced NLP apps; LangChain streamlines this process. For example, LLMs have to access large volumes of big data, so LangChain organizes these large quantities of data so that they can be accessed with ease.

In addition, GPT (Generative Pre-trained Transformer) models are generally trained on data up to their release to the public. For instance, ChatGPT was released to the public near the end of 2022, but its knowledge base was limited to data from 2021 and before. LangChain can connect AI models to data sources to give them knowledge of recent data without limitations.

LangChain typically builds applications using integrations with LLM providers and external sources where data can be found and stored. For example, LangChain can build chatbots or question-answering systems by integrating an LLM – such as those from Hugging Face, Cohere and OpenAI – with data sources or stores such as Apify Actors, Google Search and Wikipedia. This enables an app to take user-input text, process it and retrieve the best answers from any of these sources. In this sense, LangChain integrations make use of the most up- to-date NLP technology to build effective apps.

Llama Index:

LLMs offer a natural language interface between humans and data. Widely available models come pre-trained on huge amounts of publicly available data like Wikipedia, mailing lists, textbooks, source code and more.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 11, pp 603-610, November-2023

https://doi.org/10.47001/IRJIET/2023.711080

However, while LLMs are trained on a great deal of data, they are not trained on your data, which may private or specific to the problem you're trying to solve. It's behind APIs, in SQL databases, or trapped in PDFs and slide decks.

LlamaIndex solves this problem by connecting to these data sources and adding your data to the data LLMs already have. This is often called Retrieval-Augmented Generation (RAG). RAG enables you to use LLMs to query your data, transform it, and generate new insights. You can ask questions about your data, create chatbots, build semi-autonomous agents, and more.

LlamaIndex provides the following tools:

Data connectors ingest your existing data from their native source and format. These could be APIs, PDFs, SQL, and (much) more.

Data indexes structure your data in intermediate representations that are easy and perform ant for LLMs to consume.

Engines provide natural language access to your data. For example: Query engines are powerful retrieval interfaces for knowledge-augmented output. - Chat engines are conversational interfaces for multi-message, "back and forth" interactions with your data.

Data agents are LLM-powered knowledge workers augmented by tools, from simple helper functions to API integrations and more.

Application integrations tie LlamaIndex back into the rest of your ecosystem. This could be LangChain, Flask, Docker, ChatGPT, or. . . anything else!

Fine-tuning:

Fine-tuning OpenAI text generation models can make them better for specific applications, but it requires a careful in- vestment of time and effort. We recommend first attempting to get good results with prompt engineering, prompt chaining (breaking complex tasks into multiple prompts), and function calling, with the key reasons being:

There are many tasks at which our models may not initially appear to perform well, but results can be improved with the right prompts thus fine-tuning may not be necessary iterating over prompts and other tactics has a much faster feedback loop than iterating with fine-tuning, which requires creating datasets and running training jobs In cases where fine-tuning is still necessary, initial prompt engineering work is not wasted

We typically see best results when using a good prompt in the fine-tuning data (or combining prompt chaining / tool use withfine-tuning)

6. Applications of LLMs

1) Natural Language Processing

LLMs have become the default technique for NLP tasks like text classification, named entity recognition, sentiment analysis, question answering, and summarization (Brown et al., 2020). For instance, BERT achieved state-of-the-art performance on 18 NLP benchmarks (Devlin etal., 2019). LLMs provide universal representations that can be readily adapted to downstream tasks via simple fine-tuning approaches.

2) Machine Translation

LLMs have been instrumental in improving neural machine translation (NMT), surpassing traditional statistical MT models (Chen et al., 2020). Multilingual LLMs like mT5 perform translation between multiple languages using a single model, enabling efficient scaling. LLMs also allow domain specific enhancements for MT through continued pretraining on relevant corpora.

3) Content Generation

LLMs can generate high-quality text spanning news articles, reports, stories, code, and dialogue (Adiwardana et al., 2020). GPT-3 demonstrated the potential of few-shot learning for text generation across domains. LLMs are core to applications like search engines, chatbots, and creative tools.

4) Speech Recognition and Synthesis

Integrating LLMs with speech models has substantially improved automated speech recognition and text-to-speech synthesis (Baevski et al., 2020). LLMs provide the linguistic knowledge to enhance speech models' contextual understanding.

5) Information Retrieval and Extraction

LLMs advance key information retrieval tasks like document ranking, categorization, and summarization (Nogueira et al., 2019). Their contextual representations improve semantic search and question answering. For structured data, LLMs enable improved named entity recognition and relation extraction.

7. Future Work

The completion of the first phase of our project marks a significant milestone in our journey to explore the potential of

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 11, pp 603-610, November-2023

https://doi.org/10.47001/IRJIET/2023.711080

Large Language Models (LLMs) in general computer automation. It has been an exciting and enlightening experience, but the path forward is equally promising, holding numerous opportunities for further innovation and development. As we transition into the second phase, we set our sights on several avenues for future work and expansion.

- Voice Input/Output: Integrate speech recognition and synthesis for hands-free use.
- Graphical UI: Create a user-friendly graphical interface for easier interaction.
- Computer Vision: Enable generating images from descriptions using DALL-EAPI.
- Web Search: Expand document search to web pages and online sources.

8. Conclusion

The world of LLMs is a world of limitless possibilities. These language models have showcased their prowess in understanding, generating, and manipulating human language, and our project has demonstrated that their potential extends far beyond mere text-based applications. From automating code generation to streamlining data analysis and document processing, LLMs have shown that they can adapt to a multitude of tasks, making computing more accessible and user-friendly for everyone.

- Completed core modules like chatbot, document search, app opener.
- Implemented key virtual assisting capabilities for local document access, intelligent conversations etc.
- Laid the groundwork to enhance it further with voice, vision, UI features.
- Gained valuable experience in Python programming, modular & API integrated application development.

Limitations of LLMs:

Data bias: Training on limited datasets causes LLMs to perpetuate and amplify societal biases related to gender, race, culture (Bender et al., 2021).

Toxic language generation: Due to learning statistical associations, LLMs can generate harmful, biased outputs without constraint (Gehman et al., 2020).

Interpretability: The black box nature of LLMs makes it hard to explain their predictions or analyze model behavior. Lack of interpretability impedes transparency.

Resource intensity: LLMs require massive datasets and computational resources for training. Estimated training costs are upwards of $12 million for GPT-3 (Schwartz et al., 2020).

Evaluation disconnect: Benchmarks used to evaluate LLMs may not adequately test for comprehensive language understanding, generalization, and ethics (Bender and Koller, 2020).

Societal Impact: Journalism and content creation: LLMs can generate high- quality content like news articles at scale, potentially influencing information landscapes (Yuan et al., 2021).

Security and disinformation: Text generation abilities raise concerns about mass-scale disinformation generation (Zellers et al., 2019). Bias and representation: Perpetuation of societal biases impacts how LLMs represent diverse demographics and view- points (Sheng et al., 2021).

Employment: Widespread deployment of LLMs could disrupt sectors involving content creation, translation, customer service, etc. Ethics: Unethical use of LLMs for purposes like impersonation, deception, and propaganda could violate norms (Thoppilan et al., 2022).

Future Trends and Outlook:

Improved training efficiency: Approaches like knowledge distillation, noisy student training, and mixture-of-experts show promise for optimizing efficiency and scaling (Lepikhin et al., 2021; Xue et al., 2021).

Integration with symbolic AI: Combining neural LLMs with structured knowledge representation and reasoning can enhance interpretability and generalization capabilities (Lin et al., 2021).

Targeted debiasing: Techniques like data augmentation, optimized training objectives, and controlled generation help mitigate biases (Sheng et al., 2021).

Multimodal integration: Unifying LLMs with vision, speech, and robotics modalities is crucial for developing general AI systems (Agar- wal et al., 2021).

Reinforcement learning: RL-based training can enhance LLMs' grounded reasoning and exploration abilities beyond static supervised learning (Zhang et al., 2021).

REFERENCES

[1] Brown, T. B., et al. (2020). Language models are few-shot learners. arXiv preprint arXiv:2005.14165.

[2] Devlin, J., et al. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. NAACL.

ISSN (online): 2581-3048
Volume 7, Issue 11, pp 603-610, November-2023
https://doi.org/10.47001/IRJIET/2023.711080

[3] Chen, M. X., et al. (2020). A systematic survey of neural machine translation adaptation. arXiv preprint arXiv:2009.13906.
[4] Adiwardana, D., et al. (2020). Towards a human-like open- domain chatbot. arXiv preprint arXiv:2001.09977.
[5] Baevski, A., et al. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. NeurIPS.

Citation of this Article:

Priya Ethape, Riya Kane, Ghanashyam Gadekar, Sahil Chimane, "Smart Automation Using LLM" Published in *International Research Journal of Innovations in Engineering and Technology - IRJIET*, Volume 7, Issue 11, pp 603-610, November 2023. Article DOI https://doi.org/10.47001/IRJIET/2023.711080

****

