NEXUS: Neural Execution and Understanding System

Shawn Hagler
EEL-4911
Florida State University
Panama City, United States
sth20u@fsu.edu

Jaehyun Lee
EEL-4911
Florida State University
Panama City, United States
jl21bd@fsu.edu

Abstract

Leveraging the power of Large Language Models (LLMs), like ChatGPT, our enhanced virtual assistant framework transforms user interactions with its advanced comprehension and dynamic response generation. These LLMs not only unravel the complexity of human queries but also adeptly generate code, enabling the virtual assistant to seamlessly integrate and execute tasks via APIs without manual intervention. Our approach is paving the way for virtual assistants that autonomously interpret and execute a broad spectrum of complex user requests, bridging the divide between human conversation and automated action — a pivotal stride towards truly intelligent digital assistants.

I. Introduction

In the modern era, the proliferation of digital technologies has given rise to an unprecedented level of interaction between humans and machines. One of the most prevalent manifestations of this interaction is the use of virtual assistants (VAs), which have become deeply integrated into our daily lives. These assistants are called upon to perform an array of tasks, ranging from setting alarms and composing messages to conducting complex web searches. However, despite their convenience and growing capabilities, existing VA frameworks are often limited by the degree of sophistication with which they can handle and interpret human language, as well as by their ability to autonomously execute actions based on these interactions.

The crux of the challenges facing current virtual assistant technologies lies in the limitations of natural language processing (NLP) and the restricted flexibility of Application Programming Interface (API) integrations. While NLP enables machines to comprehend and process human language, it often falters when confronted with nuances, context, and the subtleties inherent in everyday communication. This shortfall is accompanied by the static nature of APIs within these systems, necessitating manual pre-coding and regular maintenance by developers to fulfill even straightforward user requests. These constraints lead to a noticeable disconnect, wherein user commands that deviate from predefined patterns or that require actions beyond the existing API integrations are either misunderstood or entirely unfulfilled.

Addressing this problem, our project explores the integration of Large Language Models (LLMs), such as ChatGPT, within the VA framework, with the aim to revolutionize the way user interactions are perceived and acted upon. The project delves into the potential of LLMs to dismantle the complexities inherent in user queries and to dynamically generate executable code. This advancement could empower VAs to directly interact with and execute tasks via APIs, reducing the reliance on manual coding and broadening the scope of autonomous functionality. The objective is to transcend the existing limitations by fostering a new generation of digital assistants that can seamlessly blend conversational comprehension with efficient task execution, thereby elevating the intelligence and utility of VAs to achieve a new benchmark in human-machine collaboration.

In our paper, we present the background information on the intersection of NLP and API interaction within the VA ecosystem, delineating the current challenges and the necessity for improvement. We then explore our approach to these challenges, offering an insightful look into how LLMs can redefine the capabilities of virtual assistants and, by extension, transform the user experience.

II. Existing Works

In the development of our project, we reviewed recent advancements within the domain of large language models and their integration with application programming interfaces (APIs). The body of work pertinent to our research encompasses two notable projects: GorillaLLM and ToolLLM, each with distinct objectives and applications in the field of machine learning.

GorillaLLM represents a step forward in language model interaction with API usage, focusing specifically on code synthesis. This system generates code sequences that enable the invocation of machine learning model APIs, with a discernible bias towards APIs facilitating other machine learning utilities such as image generation models. Despite its innovative approach to integrating API calls into language model outputs, GorillaLLM has a critical limitation: the actual execution of the code is not part of the model's pipeline. Consequently, verification of the code's functionality in real-world scenarios remains unexplored within the system's capabilities. Furthermore, its exclusive focus on machine learning model APIs restricts the breadth of applications that the model can address, leaving room for improvement and expansion to include a broader range of services.

Building upon the concept of API interaction, ToolLLM emerges as a specialized fine-tuned language model designed to optimize API selection. Unlike its predecessor, ToolLLM's primary function is to discern the most appropriate API for a given user instruction. This functionality transforms ambiguous user needs into targeted API queries, showcasing an advance in language model cognition as it pertains to understanding and utilizing various APIs across different services and platforms.

The continual progression of machine learning models, particularly those that interface with APIs, underscores the dynamic nature of the field. Our project aims to leverage and expand upon these foundational works to enhance the utility and precision of language models in code synthesis and API utilization, reflecting the ongoing evolution of machine-learning-enhanced systems and their practical applicability in software development and beyond.

As the landscape of machine learning and API integration evolves rapidly, our study is positioned to address the limitations observed in existing systems, thereby contributing to the progressive refinement of language model applications in technology.

III. Implementation

A. Large Language Model

In tackling the challenges outlined in our project, we have incorporated Large Language Models (LLMs) to provide a robust solution. These LLMs leverage the transformer neural-network architecture, which has proven highly effective for natural language processing tasks. The implementation of LLMs in our project can be depicted in two primary components: the parameter file and the run file. The success of an LLM is contingent on its neural-network which harbors a vast number of parameters. These weights embody the learned patterns from an extensive corpus of training data, enabling the model to understand and generate language effectively. The parameter file is a crucial artifact, as it stores the trained weights of the neural-network, making it possible to persist and disseminate the learned knowledge. The application of an LLM in a practical environment is facilitated by the run file. This executable code initializes the neural network, retrievs the weights from the parameter file, and orchestrates the interaction between the user's queries and the model's responses. The run file serves as a bridge connecting the pre-trained LLM with real-world applications, ensuring that the significant computational efforts invested during training are translated into tangible outputs.

For instance, we delved into the training process of the LLaMa 2 70B parameter model, a publicly accessible model released by Facebook/Meta. Starting with a colossal dataset of 10 terabytes of text, the neural network undergoes training where approximately 80% of the data is used for learning, leaving the remainder for model evaluation. The training exercise, conducted over 12 days on 6000 GPUs, culminates in a parameter file exceeding 140GB, capturing the distilled knowledge from the data.

Furthermore, the implementation phase includes provisions for fine-tuning the LLM on a domain-specific dataset. Such fine-tuning customizes the generic capabilities of the model, steering it towards a profound understanding of nuanced contexts pertinent to our project. We have demonstrated this with an example where a fine-tuned LLM exhibited a refined grasp of context, distinguishing between an article generation task and a direct question that required a response.

The fine-tuning process also brings to light the model's ability to discern the need for external resources, such as APIs, to fetch additional information for comprehensive query resolution. This capability demonstrates the LLM's potential in a wide array of applications, from AI-driven customer support to intelligent virtual assistants.

As part of our implementation, we have chosen to work with OpenAI's ChatGPT due to its state-of-the-art training with 1 trillion parameters and a rich dataset current as of April 2023. By leveraging the high-level prompting techniques, we mimic the effects of fine-tuning, directing ChatGPT to perform tasks aligned with our project needs without the computational overhead of training the model from scratch. The utilization of LLMs such as ChatGPT stands as a testament to the effectiveness of our implementation strategy. Through both pre-trained and fine-tuned models, we harness the power of advanced machine learning to address the complex challenges at the heart of our project.

B. NEXUS

In the realm of virtual assistance, our project introduces a groundbreaking system dubbed NEXUS, an end-to-end system predicated on the amalgamation of large language models and auxiliary systems formulated to eclipse existing virtual assistant technologies.

The NEXUS framework is composed of interconnected modules that orchestrate a seamless flow of operations, starting with the Natural Language Understanding (NLU) module, which serves as the cornerstone. The NLU module is adept at interpreting natural language input from users, discerning context, intent, and entities, and converting this understanding into a structured JSON-formatted query. This query initiates the interaction with the API Selector, which is the subsequent pivot in the NEXUS framework. The API Selector module is where the ingenuity of neural network technology comes into play, as it transforms the structured query into vector embeddings. These embeddings enable the system to conduct swift and semantic searches across a comprehensive API database, identifying the most relevant API or APIs that are best suited to respond to the user's specific query.

Often, queries are broad and require additional user input for accurate processing. Hence, the NEXUS system has been designed so that the NLU module might re-engage with the user to procure additional details, such as when a user instructs the system to ”Book a hotel”, the module may follow u with ”What dates?” to collect the necessary parameters fc the API’s operational requirements. With all the requisite dat in place, NEXUS can navigate towards the Code Generatc module. Here, Python code is dynamically generated base on the action schema, formulated earlier by the NLU module and inclusive of all necessary user-specified parameters. Th execution of this code bridges the user’s request with th actual API service, obtaining the sought-after result. Upo acquisition of the API’s response, the system does not ceas its operation. Instead, the response is conveyed back to the NLU module, where it is reconstituted into a natural language format, delivering an articulate and responsive answer that fulfills the user’s initial query.

NEXUS remains ever contemporary, with the API database being perpetually updated—determined by real-time data from the API hub, ensuring a relentless progression and maintenance of the system’s comprehensive capabilities. Thus. NEXUS stands as a testament to the future of virtual asstance—a sophisticated system engineered to deliver intellige context-aware, and highly interactive user experiences, sigifying a paradigm shift in how virtual assistants consumeri artificial intelligence.

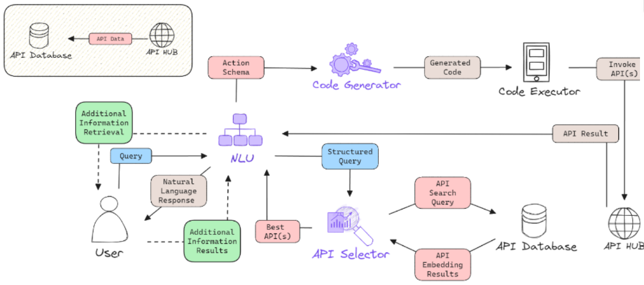
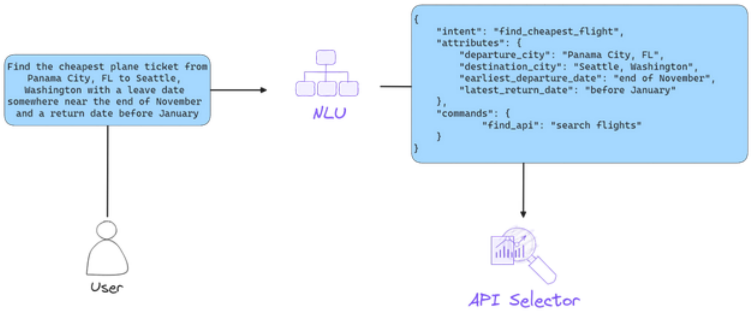


Fig. 1. Overview of NEXUS

IV. Results

In the course of our experimentation with the NEXUS system, we have conducted sample scenarios to validate the efficacy of the framework’s interconnected modules. These scenarios are designed to mimic real-world user interactions with virtual assistants and demonstrate the system’s capability to process and respond to complex queries. One such scenario involved a user request for locating the cheapest plane ticket available. The query was first processed by the NEXUS Natural Language Understanding (NLU) module, which demonstrated its ability to comprehend the request in full, evidenced by its accurate extraction of multiple attributes related to the flight search. The NLU module then generated a structured search term for the API Selector module, encapsulating the essence of the user’s intent.

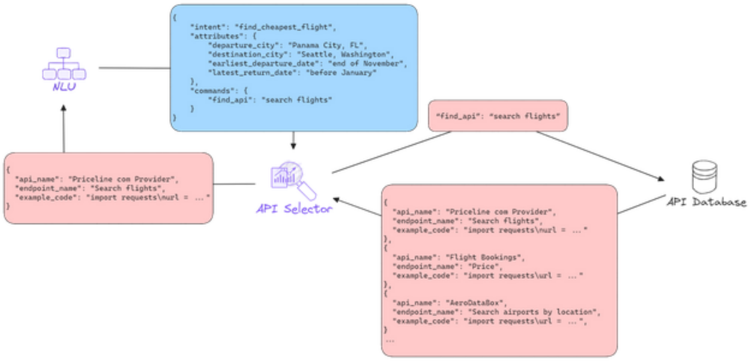
The API Selector module took center stage by interfacing with an API Database, which leveraged semantic search capabilities to retrieve multiple relevant API options. The database utilized for our example was derived from scraping a popular API marketplace, Rapid API, illustrating the flexibility and adaptability of NEXUS in real-world applications. From the queried responses, the most suitable API was identified to fulfill the user’s criteria.



Find the cheapest plane ticket from Panama City, FL to Seattle, Washington with a leave date somewhere near the end of November and a return date before January

```
{
    "intent": "find_cheapest_flight",
    "attributes": {
        "departure_city": "Panama City, FL",
        "destination_city": "Seattle, Washington",
        "earliest_departure_date": "end of November",
        "latest_return_date": "before January"
    },
    "commands": {
            "find_api": "search flights"
    }
}
```

Fig. 2. User Input to NLU



```
{
    "intent": "find_cheapest_flight",
    "attributes": {
        "departure_city": "Panama City, FL",
        "destination_city": "Seattle, Washington",
        "earliest_departure_date": "end of November",
        "latest_return_date": "before January"
    },
    "commands": {
        "find_api": "search flights"
    }
}
```

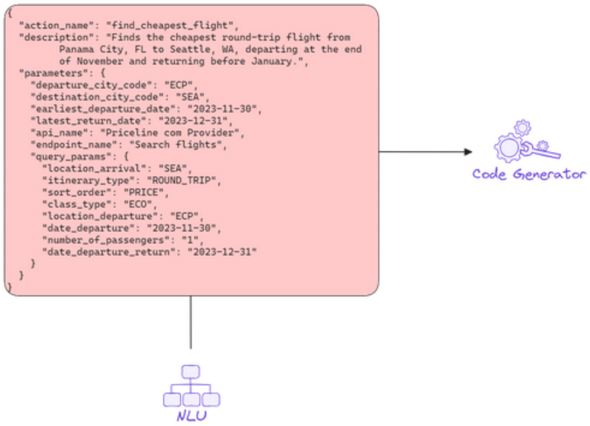
```
"find_api": "search flights"
```

```
{
  "api_name": "Priceline com Provider",
  "endpoint_name": "Search flights",
  "example_code": "import requests\nurl = ..."
}
```

```
{
  "api_name": "Priceline com Provider",
  "endpoint_name": "Search flights",
  "example_code": "import requests\nurl = ..."
},
{
  "api_name": "Flight Bookings",
  "endpoint_name": "Price",
  "example_code": "import requests\url = ..."
},
{
  "api_name": "AeroDataBox",
  "endpoint_name": "Search airports by location",
  "example_code": "import requests\url = ...",
}
...
```

Fig. 3. API Selector

Next, the NLU module formulated an action schema based on the available API information. In our scenario, this included the necessary parameters to execute the flight search, such as IATA airport codes, potential dates for departure and return, and other pertinent query parameters.

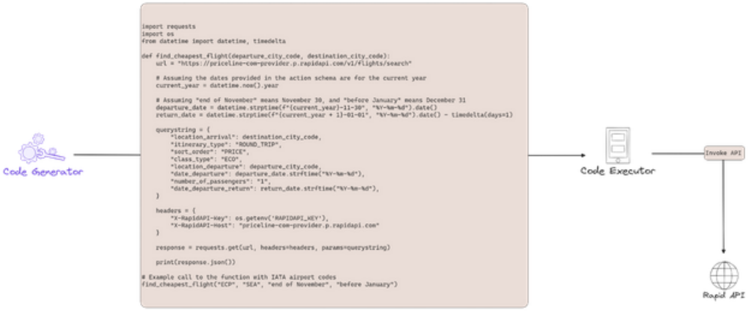


```
{
  "action_name": "find_cheapest_flight",
  "description": "Finds the cheapest round-trip flight from
        Panama City, FL to Seattle, WA, departing at the end
        of November and returning before January.",
  "parameters": {
    "departure_city_code": "ECP",
    "destination_city_code": "SEA",
    "earliest_departure_date": "2023-11-30",
    "latest_return_date": "2023-12-31",
    "api_name": "Priceline com Provider",
    "endpoint_name": "Search flights",
    "query_params": {
      "location_arrival": "SEA",
      "itinerary_type": "ROUND_TRIP",
      "sort_order": "PRICE",
      "class_type": "ECO",
      "location_departure": "ECP",
      "date_departure": "2023-11-30",
      "number_of_passengers": "1",
      "date_departure_return": "2023-12-31"
    }
  }
}
```

Fig. 4. Action Schema

The Code Generator module then synthesized executable Python code geared specifically to interact with the selected

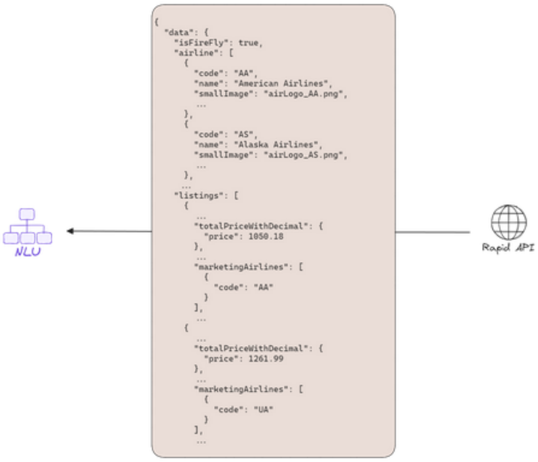
API. The code included provisions for handling requests, e vironment variables, and query parameters that conform to t user's input. This generated code represented the operation ”muscle” of NEXUS, directly engaging with external servic to procure results.



```
import requests
import os
from datetime import datetime, timedelta

def find_cheapest_flight(departure_city_code, destination_city_code):
    url = "https://priceline-com-provider.p.rapidapi.com/v1/flights/search"

    # Assuming the dates provided in the action schema are for the current year
    current_year = datetime.now().year

    # Assuming "end of November" means November 30, and "before January" means December 31
    departure_date = datetime.strptime(f"{current_year}-11-30", "%Y-%m-%d").date()
    return_date = datetime.strptime(f"{current_year + 1}-01-01", "%Y-%m-%d").date() - timedelta(days=1)

    querystring = {
        "location_arrival": destination_city_code,
        "itinerary_type": "ROUND_TRIP",
        "sort_order": "PRICE",
        "class_type": "ECO",
        "location_departure": departure_city_code,
        "date_departure": departure_date.strftime("%Y-%m-%d"),
        "number_of_passengers": "1",
        "date_departure_return": return_date.strftime("%Y-%m-%d"),
    }

    headers = {
        "X-RapidAPI-Key": os.getenv('RAPIDAPI_KEY'),
        "X-RapidAPI-Host": "priceline-com-provider.p.rapidapi.com"
    }

    response = requests.get(url, headers=headers, params=querystring)

    print(response.json())

# Example call to the function with IATA airport codes
find_cheapest_flight("ECP", "SEA", "end of November", "before January")
```

Fig. 5. Code Generator

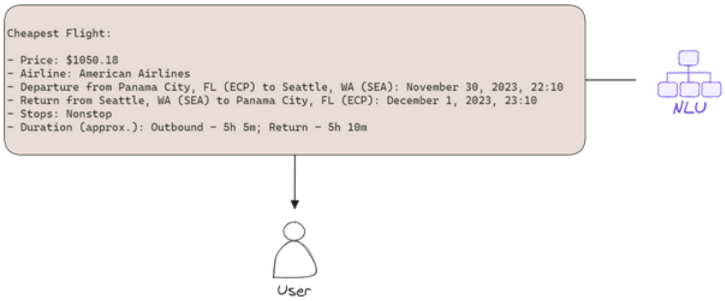
Execution of the generated code was the responsibility of the Code Executor, which interfaced with the API from Rapid API. When the API provided the response, NEXUS did not falter; it showcased its robustness by passing the resultant data back to the NLU module. This data, though raw, was processed and translated into a user-friendly format—a direct and succinct response to the user's initial query. The natural language response encapsulated detailed flight information including the lowest price found, airline details, departure and arrival information, along with additional flight specifics.



```
{
  "data": {
    "isFireFly": true,
    "airline": [
      {
        "code": "AA",
        "name": "American Airlines",
        "smallImage": "airLogo_AA.png",
        ...
      },
      {
        "code": "AS",
        "name": "Alaska Airlines",
        "smallImage": "airLogo_AS.png",
        ...
      },
      ...
    "listings": [
      {
        ...
        "totalPriceWithDecimal": {
          "price": 1050.18
        },
        ...
        "marketingAirlines": [
          {
            "code": "AA"
          }
        ],
        ...
      {
        ...
        "totalPriceWithDecimal": {
          "price": 1261.99
        },
        ...
        "marketingAirlines": [
          {
            "code": "UA"
          }
        ],
        ...
```

Fig. 6. API Result

Our results showed that NEXUS is a competent and powerful system capable of navigating through the complexities of API selection, code generation, and natural language processing to deliver actionable information to end-users. Through this experiment, NEXUS proved it can readily integrate with existing technology infrastructures, interpret user requests accurately, and facilitate automated responses in a coherent and detailed manner.



```
Cheapest Flight:

- Price: $1050.18
- Airline: American Airlines
- Departure from Panama City, FL (ECP) to Seattle, WA (SEA): November 30, 2023, 22:10
- Return from Seattle, WA (SEA) to Panama City, FL (ECP): December 1, 2023, 23:10
- Stops: Nonstop
- Duration (approx.): Outbound - 5h 5m; Return - 5h 10m
```

Fig. 7. Natural Language Response

The demonstration underpins NEXUS's potential as a transformative player in the realm of virtual assistance, affirming the system's capability to manage intricate, natural language user queries and present the resulting data with clarity and efficiency.

V. Conclusion

In summary, our study introduced and developed NEXUS, an innovative virtual assistant framework that leverages the robust capabilities of Large Language Models (LLMs) such as ChatGPT to redefine the paradigm of human-machine interaction. By integrating a sophisticated Natural Language Understanding (NLU) module, an intelligent API Selector, and a dynamic Code Generator, NEXUS successfully bridges the gap between conversational comprehension and automated task execution. Our system circumvents the traditional limitations of virtual assistant technologies by enabling direct API interactions and facilitating real-time execution of user instructions without the need for manual code intervention. Through rigorous experimentation, the NEXUS system has been empirically validated to handle complex user queries, automate the selection of appropriate APIs, generate executable code, and present actionable responses in natural language. The application of our system in practical scenarios demonstrates its proficiency in addressing real-world user needs, thereby marking a significant step forward in the evolution of virtual assistants. Moving forward, our vision for NEXUS entails the development of a proprietary LLM, the diversification of the API database, and the enhancement of the overall system's autonomy and performance. By channeling financial investment and focusing on innovation, we aim to achieve greater customization, scalability, and privacy assurance, all the while keeping apace with the rapidly changing landscape of machine learning and API development. Our project reflects a strong commitment to pushing the boundaries of what virtual assistants can achieve and sets the stage for a future where human-like digital assistants are integral and omnipresent in supporting and enriching our digital experience. NEXUS, therefore, is poised to become an archetype of the next generation of intelligent virtual assistants, marking a substantial leap in the journey towards seamless, efficient, and intelligent human-machine collaboration.

A. Future Work

Despite the successful implementation of NEXUS and promising results yielded from its evaluations, our project is poised for further advancement in the coming semesters. Current reliance on third-party language models, such as OpenAI's ChatGPT, has directed us toward the pursuit of enhanced autonomy over the tools integral to NEXUS.

A primary focus for future work is the acquisition of funds approximately amounting to $10,000. This investment is aimed at the development and fine-tuning of our own large language model (LLM). With dedicated financial resources, the project can break new ground in custom LLM training—personalizing it to better align with the specific needs of NEXUS. Ownership of the LLM would enable the team to adapt and evolve the model independently, bypassing restrictions and limitations presented by external dependencies. Cultivating our own LLM will also entail establishing a hosting solution that allows us to manage and deploy the model directly. Hosting our model would enhance the system's performance, offer increased scalability, and ensure that the privacy and security of user data are upheld to the highest standards.

In parallel with the development of an in-house language model, the expansion of the API database is another key area for growth. The intent is to curate a wider variety of APIs, diversifying the range of capabilities available to NEXUS. Special attention will be paid to the selection of APIs, ensuring each inclusion has a potential use within the virtual assistant context, hence eliminating redundancy and optimizing the system's performance. This strategy entails a meticulous vetting process that filters out duplicate or irrelevant APIs, promoting a streamlined and efficient collection that champion's utility. Our goal is to incorporate APIs that not only respond to user queries but do so with precision and relevance, taking into account the diverse and evolving demands of virtual assistant users. The continuous evolution of NEXUS will involve not only augmenting the number of APIs but also refining the integration process to foster a seamless user experience. As we proceed, our vision encompasses a NEXUS that is more robust, versatile, and tailored to the nuances of modern-day virtual assistance. Each step toward this trajectory stands as a commitment to innovation and excellence in our project.

References

[1] Patil, S. G., Zhang, T., Wang, X., and Gonzalez, J. E. (2023, May 24). Gorilla: Large Language Model Connected with Massive APIs. Retrieved from https://arxiv.org/abs/2305.15334

[2] Qin, Y. et al. (2023, October 3). ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. Retrieved from https://arxiv.org/abs/2307.16789

[3] Vaswani, A. et al. (2017, June 12). Attention Is All You Need. Retrieved from https://arxiv.org/abs/1706.03762

[4] Ouyang, L. et al. (2022, March). Training language models to follow instructions with human feedback. Retrieved from https://arxiv.org/abs/2203.02155