Noname manuscript No.
(will be inserted by the editor)

Harnessing Large Language Models for Multitasking AI Chatbot

Connor Hehn · Reyes Huerta · David McNulty · Andrew Visceglia · Dr. Sidike Paheding

Received: date / Accepted: date

Abstract The rapid growth of Artificial Intelligence (AI) in recent years has been propelled by the development of generative AI applications from industry leaders such as OpenAI, Google, and Microsoft. IBM characterizes a chatbot as a computer program simulating human conversation, with contemporary iterations increasingly integrating conversational AI techniques like Natural Language Processing (NLP). These chatbots can be utilized across diverse domains, including applications in home speakers, messenger apps, and virtual assistants. An essential factor in the progression of modern AI chatbots is the application of transformer models, exemplified by architectures like GPT-3 (Generative Pre-trained Transformer 3). These models play a pivotal role in enhancing the capabilities of conversational agents, especially through the attention mechanism, which effectively captures contextual information, contributing to more coherent and context-aware responses. This work is dedicated to leveraging the capabilities of large language models for the development of a multitasking AI chatbot that is adept at handling both voice and text commands. We implement various functionalities such as listening

Dr. Paheding
Fairfield University
E-mail: spaheding@fairfield.edu

C. Hehn
Fairfield University
E-mail: connor.hehn@student.fairfield.edu

R. Huerta
Fairfield University
E-mail: reyes.huerta@student.fairfield.edu

D. McNulty
Fairfield University
E-mail: david.mcnulty@student.fairfield.edu

A. Visceglia
Fairfield University
E-mail: andrew.visceglia@student.fairfield.edu

to users, engaging in naturalistic conversations, providing maps or directions for navigational purposes, and playing music, among others. Beyond these advanced features, the chatbot boasts language proficiency, enabling it to understand and respond in five languages. This multilingual capability significantly enhances the chatbot's accessibility and usability, catering to a broader user base. Ultimately, we intend to define the chatbot's desired capabilities and fine-tune existing language models to achieve the intended results.

Keywords Artificial Intelligence · WebApp · Flask

1 Introduction

The project is dedicated to leveraging the capabilities of large language models for the development of a multitasking AI chatbot that is adept at handling both voice and text commands. Furthermore, the implementation includes various functionalities such as listening to users, engaging in naturalistic conversations, providing maps or directions for navigational purposes, and playing music. Beyond these advanced features, the chatbot boasts multilingual proficiency, enabling it to understand and respond in English, Spanish, Italian, French, and German. The multilingual capability significantly enhances the chatbot's accessibility and usability, catering to a broader user base. The chatbot's desired capabilities were fine-tuned using existing language models.

The overarching objective of this project was to deliver an accessible and multitasking AI chatbot application, specifically tailored to meet the needs of the Fairfield University community, in particular, first-year students and prospective students. Ultimately, the significance of transformer models in shaping the landscape of modern AI chatbots cannot be overstated, as they empower these systems to provide more sophisticated and contextually aware conversational experiences.

2 Related Work

Previous research has shown significant advancements in Artificial Intelligence Chatbots, particularly working with large language models (LLM). Many companies such as OpenAI, Microsoft, Meta and Google have developed language models that are used in the field of AI. OpenAI's model GPT-3 (Generative Pre-trained Transformer) uses a transformer model to produce human-like text [9]. Models like the one developed by OpenAI "write automatically and autonomously texts of excellent quality, on demand" [9]. An additional model developed by Google is called BERT (Bidirectional Encoder Representations from Transformers). BERT is a state of the art model built for a "range of tasks, such as question answering and language inference" [8]. Needless to say there are many companies involved in the artificial intelligence field building language models using transformers. The consequences of these advancements cannot be understated, “it is the biggest transformation of the writing process since the word processor” [9].

Products utilizing language models are becoming more available as advancements continue to be made. One recent product such as the Rabitt R1 uses AI in this sense. The product is described as “standalone AI device” [11]. This device has the capability to “control your music, order you a car, buy your groceries, send your messages, and more” [11]. The ability to control apps is due to the creation of the Large Action Model (LAM), “trained by humans interacting with apps like Spotify and Uber, essentially showing the model how they work.” [11]. Artificial Intelligence has grown significantly in the past years and some of the “most significant consequences are already imaginable” [9].

By leveraging existing language model technology and fine tuning them for the specific requirements of our project, we aim to provide a unique solution to Fairfield University students.

3 Methodology

The methodology of building the web application includes various tools such as APIs, language models, multiple programming languages, and the Flask framework. Flask is a lightweight python framework useful for building web applications [1]. Python is the primary programming language used for server side logic [7]. The Flask framework allows the creation of RESTful APIs using python. Flask was used with other various libraries to help achieve the desired multitasking functionality, such as speech recognition, mapping, and music capabilities. The architecture for the application can be seen in Figure 1. The figure showcases how the specific modules interact with each other. The specific models for each functionality are outlined in the following sections.

3.1 Language Model

The large language model (LLM) was specifically chosen from Hugging Face with the intention of being fast, accurate, and free. Among various language models tested, the language model that was ultimately chosen was the Mixtral-8x7B. This model is a pre-trained generative Sparse Mixture of Experts, which follows the GPT (Generative Pre-trained Transformer) architecture, but uses a sparse mixture of experts that enhances its capabilities beyond a standard GPT model [10]. The model “outperforms Llama 2 70B on most benchmarks” [10]. This model was incorporated into the web application using Hugging Face Hub endpoints [3]. Hugging Face Hub allows easy interaction between python code and language models on Hugging Face such as Mixtral-8x7B.

To achieve proper functionality with the language model using Hugging Face endpoints, there are multiple methods used to correctly format the input and output of the conversation. In addition, there are also multiple parameters

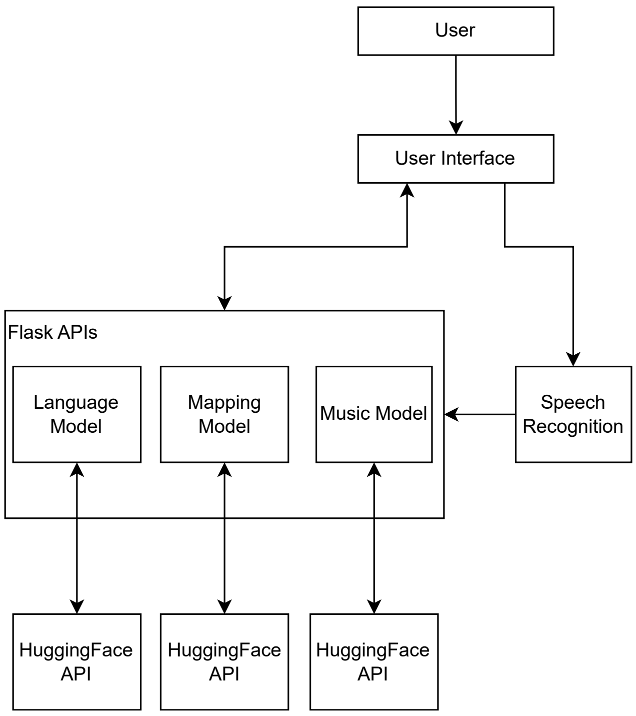


Fig. 1 High Level Architecture of Application.

used by the model such as: system prompt, temperature, max new tokens, top-p (nucleus sampling), and repetition penalty. System prompt is a string passed to the language model which provides the initial context for the conversation. In the case of this application, the model was prompted with the instructions that it is a virtual assistant, located at Fairfield University. This context helps the model answer questions related to Fairfield University. The system prompt is helpful in creating the language selector functionality in the application. By having the same system prompt in multiple languages, the application will use the system prompt in the language selected by the user, thus producing an output in that same language selected. The temperature parameter determines the randomness of the responses. The max tokens parameter limits the size of the response from the model. Top-p (nucleus sampling) is a parameter in which a higher value produces more low-probability tokens. This helps in generating diverse yet comprehensive outputs. The final parameter is the repetition penalty, which determines how the model penalizes tokens that have already appeared in the output. A higher value leads to more diverse outputs. The parameters listed above provide the unique context for the application built, and provide a unique user experience to the end users, specifically designed for use at Fairfield University.

3.2 Music Model

In order to achieve our desired music functionality we utilized the Spotify Web API in our multitasking AI chatbot. Access to Spotify was achieved through a free developer account which granted us an access token required to authenticate our API requests. The OAuth 2.0 is an industry-standard for authorization, and ensures that only those with express access to the chatbot are able to be properly authenticated. Spotify's API was a clear standout among competitors; the extensive music catalog and wide search engine made it a perfect match for our desired music functionality.

The API is prompted and requested through our application's Flask endpoints. Spotify requests are separated into two routing functions, one to handle the music search and another to perform the audio playback. Returned parameters about the songs include track name, album name, artist, album image, and finally the preview URL which contains the 30-second song preview.

To access the music functionality, users are able to select the Spotify button on the left hand navigation. Once an artist name or song is searched, the top five closest results are returned. Users can select their desired track to play a quick preview.

3.3 Mapping Model

The mapping functionality for the chatbot application was achieved through Python code that leveraged the GraphHopper API for geocoding and routing. GraphHopper is an open-source routing engine built on top of OpenStreetMap (OSM) data that is primarily designed for solving complex routing problems for several modes of transportation (driving, walking, biking, etc.)[6]. By providing powerful tools to integrate routing capabilities into web and mobile applications, GraphHopper enables users to obtain precise directions and optimized routes in an efficient manner. The geocoding API converts textual addresses into geographic coordinates (latitude and longitude). This is useful for finding exact locations based on user-provided address data. The parameters typically include the query string and an API key. The routing API calculates routes between given coordinates for various forms of transport. In this case, the desired mode of transportation was walking. The API returns detailed route information including distance, estimated time, and step-by-step directions. Another key benefit of this API is its support for polyline encoding of the route path, which can be decoded to display routes on maps. The parameters include the starting point, vehicle type, and API key.

The following libraries were required to achieve the desired mapping functionality: requests (for making HTTP requests)[5], folium (for creating interactive maps)[2], polyline (for decoding polylines returned by the routing API into coordinate lists)[4], and ipywidgets as well as IPython.display (for creating and displaying interactive widgets). The geocoding function was created in order to make a geocoding request to the GraphHopper API. It constructs a URL for the GraphHopper geocoding endpoint, sends a GET request, and parses the response JSON to extract latitude and longitude coordinates of the result. Moreover, dropdown widgets were created using ipywidgets to allow the user to select start and end locations from a predefined list of locations on Fairfield University's campus. The locations are sorted alphabetically for easier user interaction. Furthermore, a button for triggering the route calculation and an output widget to display results (interactive map, distance, time, and directions) were set up. When the user clicks the button, the route calculation function is called. This function retrieves the selected start and end locations from the dropdowns, calls the geocoding function to obtain coordinates for these locations, constructs a routing URL using these coordinates and makes a GET request to the GraphHopper routing API, and parses the routing API response to display the total distance, time, and step-by-step walking directions. Ultimately, an interactive map is generated showing the route with markers for the start and end points and a line tracing the route based on the decoded polyline.

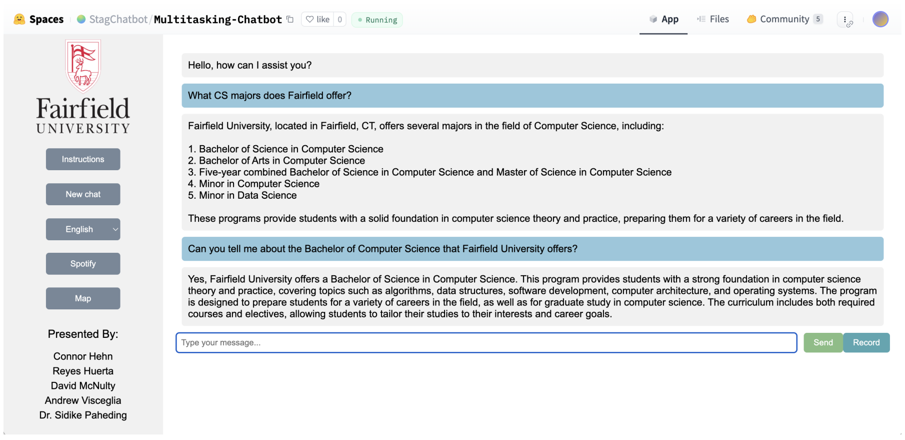
GraphHopper competes with several other routing engines like OSRM (Open Source Routing Machine), Google Maps, and Mapbox Directions; however, GraphHopper allows for customization and integration without the high costs associated with its competitors. The decision to select GraphHopper to achieve the desired mapping functionality came down to its cost (entirely free) as well as its ability to provide fast and reliable walking directions for navigating Fairfield University's campus.

3.4 Speech Recognition Model

The speech recognition functionality for the application enables users to seamlessly interact with the application using spoken commands. This feature enhances user functionality and accessibility. The model chosen for speech recognition is the webkitSpeechRecognition which is part of the Web Speech API provided by modern web browsers. When a user presses the record speech button, a new speech recognition object is created. Based on specific handlers, the spoken text is then inputted into the text area and automatically submitted to the chatbot for processing. This model is not implemented in the Flask back-end written with Python, instead it is written in JavaScript. This methodology allows for faster processing and a shorter delay when using the speech recognition model.

4 Results and Discussion

Our chatbot web application, titled StagChatbot, provides a flexible interface for interaction. Users can engage in discussions or ask any question by simply typing into the text box, ensuring a seamless communication. As an alternative option, they can explore preset questions specifically related to Fairfield University, catering to a more tailored experience. To enhance accessibility, users



Hello, how can I assist you?

What CS majors does Fairfield offer?

Fairfield University, located in Fairfield, CT, offers several majors in the field of Computer Science, including:

1. Bachelor of Science in Computer Science
2. Bachelor of Arts in Computer Science
3. Five-year combined Bachelor of Science in Computer Science and Master of Science in Computer Science
4. Minor in Computer Science
5. Minor in Data Science

These programs provide students with a solid foundation in computer science theory and practice, preparing them for a variety of careers in the field.

Can you tell me about the Bachelor of Computer Science that Fairfield University offers?

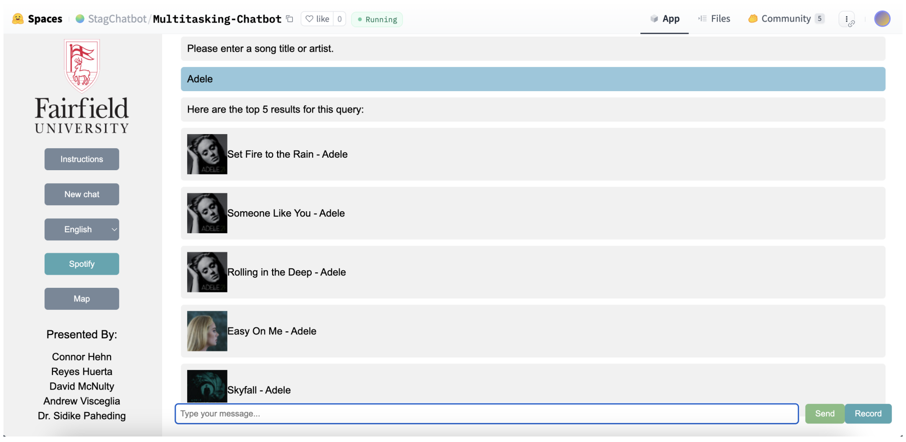
Yes, Fairfield University offers a Bachelor of Science in Computer Science. This program provides students with a strong foundation in computer science theory and practice, covering topics such as algorithms, data structures, software development, computer architecture, and operating systems. The program is designed to prepare students for a variety of careers in the field, as well as for graduate study in computer science. The curriculum includes both required courses and electives, allowing students to tailor their studies to their interests and career goals.

Presented By:

Connor Hehn
Reyes Huerta
David McNulty
Andrew Visceglia
Dr. Sidike Paheding

Fig. 2 Conversational usage of the application.

also have the option to switch the language to their preferred native language all while promoting inclusively and comfort in communication. Languages included are English, Spanish, German, Italian, and French. Furthermore, our speech recognition feature enables users to engage in an immersive text-to-speech conversation with our AI chatbot, creating a dynamic and interactive experience. Whether typing or speaking, our chatbot aims to facilitate seamless communication and engagement for all of our users. This functionality can be seen in Figure 2.



Please enter a song title or artist.

Adele

Here are the top 5 results for this query:

Set Fire to the Rain - Adele

Someone Like You - Adele

Rolling in the Deep - Adele

Easy On Me - Adele

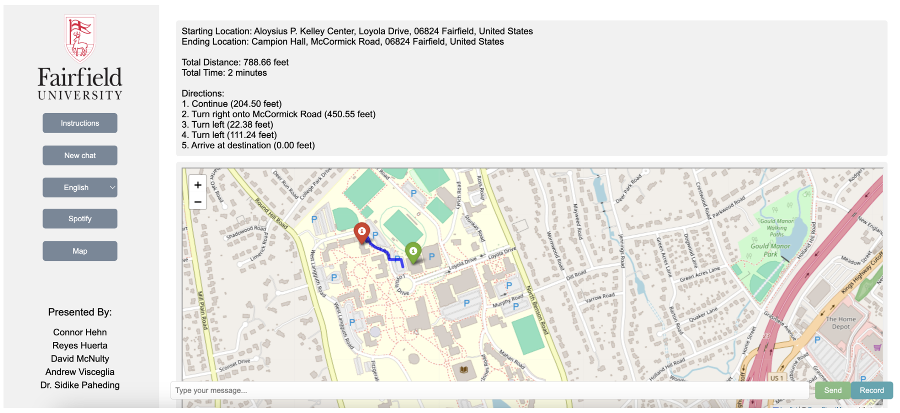
Skyfall - Adele

Presented By:

Connor Hehn
Reyes Huerta
David McNulty
Andrew Visceglia
Dr. Sidike Paheding

Fig. 3 Music functionality of the application.

When navigating our web application, the user will be able to locate the Spotify functionality that's positioned in the left-hand navigation bar. Upon selecting the Spotify feature, users will then be asked by the chatbot to identify what song title or artist they would like to listen to. For example, when a user types in “Adele” as the artist, the chatbot will then provide the top 5 Adele songs. Once the top 5 songs are listed, the user may select a song they would like and listen to a 30-second snippet that would give the user a taste of what the song is about. But, it’s also important to keep in mind that, even if our Spotify feature improves the user experience, the preview URLs might not be available for every song. However, this functionality offers consumers a smooth and entertaining way to explore music within our web application. This functionality can be seen in Figure 3.


Starting Location: Aloysius P. Kelley Center, Loyola Drive, 06824 Fairfield, United States
Ending Location: Campion Hall, McCormick Road, 06824 Fairfield, United States

Total Distance: 788.66 feet
Total Time: 2 minutes

Directions:
1. Continue (204.50 feet)
2. Turn right onto McCormick Road (450.55 feet)
3. Turn left (22.38 feet)
4. Turn left (111.24 feet)
5. Arrive at destination (0.00 feet)

Presented By:
Connor Hehn
Reyes Huerta
David McNulty
Andrew Visceglia
Dr. Sidike Paheding

Fig. 4 Mapping functionality of the application.

Right underneath our Spotify functionality, the user can also locate the mapping functionality to seamlessly navigate throughout our university campus. This feature works exceptionally well with any prospective student or first-year student trying to navigate through campus for the first time. When selecting the mapping functionality, the user will be prompted with two list selections. The first list selection is for the user to select the starting location where the user is currently while the second list selection is for the user to select the ending location where the user wants to go. In Figure 4, we can see how the user selected The Kelley Center (where admissions is located) as its starting location and the Campion Hall (a first year dorm building) as the ending location. Once the user has the starting and ending locations set, then the chatbot will give a written explanation which includes the total distance, total time, and directions in feet of the travel to the desired location as well as a visual map that shows the user the route with both the starting location highlighted in green and the ending location highlighted in red. Finally, for a better-detailed experience, the user will be able to see the names of every building, facility, and field on campus for a more immersive and user-friendly experience. This functionality can be seen in Figure 4.

5 Conclusion and Future Work

The final version of the StagChatbot achieved all of its desired functionality. StagChatbot is the first chatbot of its kind for the Fairfield University community; the application was completely built from the ground up. Currently, the application is being hosted as a web application on Hugging Face Spaces at the following link:

https://huggingface.co/spaces/StagChatbot/Multitasking-Chatbot

The graphical user interface (GUI) includes buttons for the user to select the "Instructions" for using the chatbot, starting a "New Chat," navigating to either the "Spotify" or "Map" functionality, and a drop down select list for choosing a language to interact with the chatbot (English, Spanish, Italian, French, or German). The user has the option to manually type in their message to the chatbot, select one of the pre-existing prompts, or record their speech. Ultimately, the language model was pre-prompted to expect user input regarding Fairfield University in order to ensure heightened accuracy in its generated results. Furthermore, the user can obtain detailed step-by-step walking directions from one location to another on Fairfield University's campus, along with the total time and distance required to reach their destination. Another key feature of the mapping functionality is the interactive map that is returned with the previously aforementioned output. This map indicates the start and end locations with color-coded pins as well as the path the user will follow. Moreover, the Spotify functionality enables the user to request either an artist or song title. The returned results will consist of the five options that most closely match the user's request. The user has the option of playing a 30-second sample of the song they requested.

There are several enhancements that future versions of StagChatbot hope to have employed. First off, the current version of StagChatbot does not store any user data. In the future, a database could be connected to the application that enables users to create their own account and interact with the chatbot concurrently without interference. This would require the knowledge of a cybersecurity expert to help ensure that user data is being handled safely and ethically. Another enhancement would be eventually transitioning the chatbot from a web application to an iOS or Android application. This would allow for users to more easily access the chatbot on their mobile devices and tablets. An alternative to creating a new mobile application would be integrating the chatbot with the pre-existing "Fairfield U" app. Additionally, future versions of the chatbot could include brand-new functionality and/or updates to pre-existing functionality (mapping, Spotify, etc.). Ultimately, the language model can be further enhanced by training it on more data pertaining to Fairfield University and the greater Fairfield community.

Acknowledgements The successful completion of this project would not have been possible without supervision of Dr. Sidike Paheding, guidance from Professor Akshay Mathur, and support from the Fairfield University School of Engineering and Computing.

References

1. Flask documentation. https://flask.palletsprojects.com/en/2.1.x/. Accessed: May 2024
2. Folium documentation. https://python-visualization.github.io/folium/. Accessed: May 2024
3. Hugging face hub documentation. https://huggingface.co/docs/hub/. Accessed: May 2024
4. Polyline documentation. https://pypi.org/project/polyline/. Accessed: May 2024
5. Python requests documentation. https://docs.python-requests.org/en/latest/. Accessed: May 2024
6. Graphhopper documentation (2017). URL https://www.graphhopper.com/
7. Python documentation (2024). URL https://www.python.org/
8. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
9. Floridi, L., Chiriatti, M.: Gpt-3: Its nature, scope, limits, and consequences. Minds and Machines **30**, 681–694 (2020)
10. Jiang, A.Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D.S., de las Casas, D., Hanna, E.B., Bressand, F., Lengyel, G., Bour, G., Lample, G., Lavaud, L.R., Saulnier, L., Lachaux, M.A., Stock, P., Subramanian, S., Yang, S., Antoniak, S., Scao, T.L., Gervet, T., Lavril, T., Wang, T., Lacroix, T., Sayed, W.E.: Mixtral of experts (2024)
11. Pierce, D.: The rabbit r1 is an ai-powered gadget that can use your apps for you (2024). URL https://www.theverge.com/2024/1/9/24030667/rabbit-r1-ai-action-model-price-release-date