
DEVELOPMENT OF A CONVERSATIONAL LANGUAGE MODEL FOR AUTOMATED AIRLINE TICKET BOOKING

Rustam Akimov¹

¹RTU MIREA University

Abstract

This paper discusses the creation of a conversational language model for automated airline ticket booking. Utilizing the large language model Mistral-7B-Instruct-v0.1, the system interacts with users to book flights. Flight data is sourced from a pandas DataFrame and stored in a ChromaDB vector database. Upon booking, user and ticket information is saved back into the database. This work showcases the potential of AI in streamlining complex tasks in the travel industry.

Keywords: LLM, Vector DB, BERT

1 INTRODUCTION

The advent of artificial intelligence (AI) has revolutionized various sectors, including the travel industry. One of the significant advancements in this field is the development of conversational language models that can automate complex tasks such as airline ticket booking. This paper presents a system that leverages the capabilities of a large language model, Mistral-7B-Instruct-v0.1, to interact with users and book flights.

The system is designed to understand and process user requests in a conversational manner, thereby providing a user-friendly interface for airline ticket booking. The flight data is sourced from a pandas DataFrame, a data structure that allows efficient data manipulation and analysis. This data is then stored in a ChromaDB vector database, a high-performance database system that enables efficient storage and retrieval of data.

Upon booking, the user's ticket information, along with their personal details, is saved back into the vector database. This process not only ensures the preservation of booking details for future reference but also allows for easy retrieval and modification of booking information.

This work aims to demonstrate the potential of AI in automating complex tasks, thereby streamlining processes in the travel industry. The following sections will delve into the details of the system's design and operation, highlighting its efficiency and user-friendly nature.

1.1 System Architecture

The architecture of our system is designed to facilitate efficient interaction with users and streamline the process of airline ticket booking. It comprises three main components: the language model, the database system, and a user name extraction module powered by bert-large-NER.

The language model, Mistral-7B-Instruct-v0.1, is at the heart of our system. It is responsible for understanding user requests, processing them, and generating appropriate responses. This model is designed to handle natural language inputs, making the system user-friendly and accessible to a wide range of users.

The user name extraction module utilizes bert-large-NER, a fine-tuned BERT model that is ready to use for Named Entity Recognition (NER). This model achieves state-of-the-art performance for the NER task, allowing us to accurately identify and extract user names from the conversational inputs.

The database system, ChromaDB, is used to store and retrieve flight data. Initially, the flight data is stored in a pandas DataFrame, which allows for efficient data manipulation and analysis. This data is then transferred to the ChromaDB vector database for long-term storage and easy retrieval.

Together, these components form a robust system that can automate the complex task of airline ticket booking, while providing a personalized user experience. The following sections will provide a detailed discussion of these components and their roles in the system.

2 DETAILED COMPONENTS DESCRIPTION

This section provides an in-depth look at the key components of our system: the Mistral-7B-Instruct-v0.1 language model, the bert-large-NER for user name extraction, and the ChromaDB database system. Each component plays a vital role in processing user requests, extracting necessary information, and managing flight data, collectively ensuring a seamless and efficient airline ticket booking process.

2.1 *Large Language Model*

The core of our system is the Mistral-7B-Instruct-v0.1 Large Language Model (LLM), a state-of-the-art generative text model pre-trained with an impressive 7 billion parameters. This vast number of parameters allows the model to capture a wide range of linguistic patterns and nuances, contributing to its superior performance in handling natural language processing tasks. This model outperforms Llama 2 13B on all benchmarks tested, demonstrating its high efficiency and effectiveness.

Mistral-7B-Instruct-v0.1 is a transformer model, a type of model that has revolutionized the field of natural language processing with its ability to capture long-range dependencies in text. This model incorporates several architectural choices to enhance its performance:

Grouped-Query Attention: This feature allows the model to focus on different parts of the input when generating each part of the output, improving the coherence and relevance of the generated text.

Sliding-Window Attention: This mechanism enables the model to handle longer texts by focusing on a sliding window of the most recent inputs, making it more efficient and scalable.

Byte-fallback BPE tokenizer: This tokenizer breaks down the input text into subword units, allowing the model to handle a wide range of words and languages, including those not seen during training.

In our system, we utilize the Q4_K_M quantized version of this model, formatted in GGUF. This version maintains the high performance of the original model while reducing its size and computational requirements, making it more efficient for deployment in practical applications.

Mistral-7B-Instruct-v0.1 is Mistral AI's first Large Language Model. It is trained on massive amounts of data and is capable of generating coherent text and performing various natural language processing tasks. This makes it an ideal choice for our system, which requires the ability to understand and generate natural language in order to interact with users and book airline tickets.

In addition to the vast amount of data it was trained on, the LLM also utilizes specific information from a flight database with system context. This allows the language model to seamlessly transition

into the role of an assistant, being pre-informed about certain aspects. This integration of specific data and system context enhances the model’s ability to provide accurate and relevant responses, further improving its effectiveness as an assistant.

2.2 User Name Extraction

The bert-large-NER is a fine-tuned BERT model that we use for Named Entity Recognition (NER), a crucial task in information extraction. Specifically, we use this model to extract user names from the text-based user requests. The bert-large-NER model is designed to identify and categorize key information in a text, such as names of people, organizations, locations, expressions of times, quantities, and other relevant data.

In the context of our system, the bert-large-NER model plays a vital role in personalizing the user experience. By accurately identifying and extracting user names from the conversational inputs, the system can address users by their names, making the interaction more personal and engaging.

2.3 Database System

ChromaDB is the database system we use for storing and retrieving flight data. Initially, the flight data is stored in a pandas DataFrame, a data structure that allows for efficient data manipulation and analysis. This data is then transferred to the ChromaDB vector database for long-term storage and easy retrieval.

ChromaDB is a high-performance database system that enables efficient storage and retrieval of data. It is particularly suited for handling large volumes of data, making it an ideal choice for our system, which needs to manage extensive flight data. Upon booking, the user’s ticket information, along with their personal details, is saved back into the vector database. This process not only ensures the preservation of booking details for future reference but also allows for easy retrieval and modification of booking information.

2.4 User Information Extraction Methods

In addition to the user name extraction, our system also employs a series of methods to extract other essential user information from the text-based user requests. This information includes dates, birth dates, email addresses, numbers, classes of service, gender, city, and specific values.

We use a combination of regular expressions, date parsing, and natural language processing techniques to extract this information. For instance, we use the dateutil library’s parse function to extract dates from the text, and regular expressions to extract email addresses and numbers. We also use a list of synonyms to identify the user’s gender and the class of service they prefer for their flight.

For more complex tasks, such as extracting names, we use the previously mentioned bert-large-NER model, which is specifically designed for Named Entity Recognition. This model allows us to accurately identify and extract names from the conversational inputs.

These methods enable us to extract all the necessary information from the user’s text request, allowing the system to process the request and book the airline ticket accurately and efficiently.

3 SYSTEM WORKFLOW

Our system, powered by the Mistral-7B-Instruct-v0.1 language model, primarily operates in two modes:

1. **Ticket Booking:** In this mode, the system assists the user in booking an airline ticket. The user is provided with information about flight routes, departure times, and prices in a conversational manner. The flight information is sourced from a pandas DataFrame, which allows for efficient data manipulation and analysis. The language model helps the user navigate through this information and select a suitable flight. Once the user has chosen a flight, their details, along with the ticket information, are stored in a ChromaDB vector database. This mode involves various text processing methods, including the use of the bert-large-NER model for user name extraction.
2. **Ticket Retrieval:** In this mode, the system retrieves the user’s booked tickets based on a text request. The user provides a description of the ticket (including their name, departure date, arrival date, etc.), and the system searches the vector database for matching tickets. The text request is fed into the database in its raw form, and the database returns any tickets that match the user’s description.

3.1 Synthetic Flight Data Generation

To simulate the process of booking airline tickets, our system uses synthetic flight data. This data is generated using a Python script that creates random flight details, including the city of destination, departure and arrival dates, seat number, and ticket price.

The flight data is structured in a pandas DataFrame, with each row representing a unique flight. Here is a sample of the generated data.

city_name	departure_date	arrival_date	seat_place	price
Rostov	2023-07-12 21:00:00	2023-07-13 00:00:00	E25	889
Ufa	2023-09-02 23:00:00	2023-09-03 02:00:00	D15	1360
Ufa	2023-01-22 07:00:00	2023-01-22 10:00:00	B15	1481
Yekaterinburg	2023-06-28 03:00:00	2023-06-28 06:00:00	C2	1402
Novosibirsk	2023-05-09 23:00:00	2023-05-10 02:00:00	E23	947

Table 1. Sample of the generated synthetic flight data

The generation script uses a list of city names and randomizes the other details to create a diverse set of flight data. The departure and arrival dates are generated within a specified range, and the seat number is a combination of a random letter (representing the row) and a random number (representing the seat in the row). The ticket price is a random number within a specified range.

This synthetic data allows us to test the functionality of our system in a controlled environment, ensuring that it can handle a wide range of scenarios and user requests.

3.2 User Ticket Information

Once a user has booked a ticket, the system stores the ticket information along with the user’s details. This information is crucial for retrieving the ticket later and for providing a personalized user experience.

The ticket information includes details about the flight (such as the city of destination, departure and arrival dates, seat number, and ticket price) and details about the user (such as their name, document number, gender, birth date, and email address).

Here is an example of a booked ticket with the user’s information:

Ticket Info	Value
city_name	Perm
ticket_id	8
departure_date	2023-11-21 08:00:00
arrival_date	2023-11-21 11:00:00
seat_place	B30
price	804
class_of_service	economy
user_name	Ivan Ivanov
document_number	8212 877123
gender	male
birth_date	2003-01-30 00:00:00
email	iuser@ya.ru

Table 2. Sample of a booked ticket with user information

This table represents a booked ticket for a user named Ivan Ivanov, who is traveling to Perm. The ticket details and the user’s personal information are stored together, allowing for easy retrieval and verification.

3.3 Vector Database for Textual Queries

ChromaDB is a vector database system that we use for storing and retrieving flight data. It is particularly suited for handling large volumes of data and for performing complex search operations based on textual queries.

The key feature of ChromaDB is its ability to vectorize textual queries. When a user provides a description of a ticket, the system converts this text into a vector using a text embedding model. For this purpose, we use the princeton-nlp/sup-simcse-roberta-large model, which is known for its high performance in creating meaningful and contextually rich text embeddings.

Once the text is vectorized, ChromaDB calculates the cosine distances between this vector and the vectors of the tickets stored in the database. The cosine distance is a measure of similarity between two vectors, with smaller distances indicating higher similarity. By calculating the cosine distances, the system can identify the tickets that most closely match the user’s description.

This process of vectorizing textual queries and calculating cosine distances allows ChromaDB to efficiently and accurately retrieve tickets based on a user’s text request. This makes it an ideal choice for our system, which needs to manage extensive flight data and handle complex user requests.

4 DISCUSSION

In this section, we discuss the strengths and limitations of our system, as well as potential improvements for future iterations.

4.1 Strengths and Limitations

The use of the Mistral-7B-Instruct-v0.1 language model in our system presents significant potential. It allows for a conversational interaction with the user, making the process of booking airline tick-

ets more intuitive and user-friendly. However, like all Large Language Models (LLMs), it has its limitations.

One of the main challenges with LLMs is their tendency to hallucinate, or generate false information. In the context of our system, this can manifest as the generation of incorrect flight times or ticket prices. Despite its impressive performance, the Mistral-7B-Instruct-v0.1 model, with its 7 billion parameters, sometimes fails to adhere to the system context and generates incorrect facts.

4.2 Future Improvements

To address the limitations of the current system, future iterations could consider using a larger language model. Models with 70 billion parameters (like Llama-2-70b) or even over 100 billion parameters (like Falcon-180B) could potentially provide more accurate and contextually appropriate responses.

Another potential improvement could be to move away from the quantized version of the model. While quantization helps reduce the size and computational requirements of the model, it may also contribute to the generation of incorrect information. Using the full version of the model could help mitigate this issue and improve the overall performance of the system.

5 CONCLUSION

This project has demonstrated the potential of leveraging artificial intelligence, specifically large language models like Mistral-7B-Instruct-v0.1, in automating complex tasks such as airline ticket booking. The system we've developed is capable of understanding and processing user requests in a conversational manner, providing a user-friendly interface for airline ticket booking.

The flight data, sourced from a pandas DataFrame, is efficiently managed and stored in a ChromaDB vector database. This high-performance database system enables efficient storage and retrieval of data, ensuring the preservation of booking details for future reference and allowing for easy retrieval and modification of booking information.

However, like all AI systems, ours is not without its limitations. The tendency of large language models to hallucinate, or generate false information, is a challenge we've encountered. Future improvements to our system could include the use of larger language models or moving away from the quantized version of the model to mitigate this issue.

In conclusion, our work has shown that AI can significantly streamline processes in the travel industry, making tasks like airline ticket booking more efficient and user-friendly. As AI continues to evolve, we can expect to see even more improvements in this field, further enhancing the user experience and the efficiency of operations.

REFERENCES

- [1] Microsoft Research AI4Science, & Microsoft Azure Quantum. (2023). The Impact of Large Language Models on Scientific Discovery: a Preliminary Study using GPT-4. *arXiv preprint arXiv:2311.07361*. <https://arxiv.org/abs/2311.07361>
- [2] Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J-Y., & Wen, J-R. (2023). A Survey of Large Language Models. *arXiv preprint arXiv:2303.18223*. <https://arxiv.org/abs/2303.18223>
- [3] Birhane, A., Kasirzadeh, A., Leslie, D., & Wachter, S. (2023). Science in the age of large language models. *Nature Reviews Physics*. <https://www.nature.com/articles/s42254-023-00581-4>
- [4] Taulli, T. (2023). Large Language Models. In *Springer*. https://link.springer.com/chapter/10.1007/978-1-4842-9367-6_5
- [5] Makridakis, S., Petropoulos, F., & Kang, Y. (2023). Large Language Models: Their Success and Impact. *MDPI*. <https://www.mdpi.com/2571-9394/5/3/30>
- [6] Monigatti, L. (2023). From prototype to production: Vector databases in generative AI applications. *Stack Overflow Blog*. <https://stackoverflow.blog/2023/10/09/from-prototype-to-production-vector-databases-in-generative-ai-applications/>
- [7] (2023). Exploring Vector Databases: A Guide to Their Role in AI Tech. *ProjectPro*. <https://www.projectpro.io/article/vector-databases/903>
- [8] Okeh, O. (2023). How to implement a vector database for AI. *LogRocket Blog*. <https://blog.logrocket.com/implement-vector-database-ai/>
- [9] (2023). Exploring Vector Databases: A Guide to Their Role in AI Tech. *ProjectPro*. <https://www.projectpro.io/article/vector-databases/903>
- [10] Birhane, A., Kasirzadeh, A., Leslie, D., & Wachter, S. (2023). Science in the age of large language models. *Nature Reviews Physics*. <https://www.nature.com/articles/s42254-023-00581-4>