FRI Natural Language Processing Course 2025
Project report

University *of Ljubljana*
Faculty *of Computer and*
Information Science

Coccodrillo: A Retrieval-Augmented Conversational Agent for Smart Travel Guidance

Davide Belcastro, Constance Monluc, Ondra Pritel

Abstract

Coccodrillo is a multilingual, retrieval-augmented chatbot engineered to assist users in planning safe and informed travel experiences. Unlike traditional chatbots, it combines real-time data retrieval from web sources with curated datasets to answer questions about local safety, weather alerts, cuisine, events, transportation, and sightseeing. The system integrates a compact BERT-based intent classifier, a QA module for entity resolution, and a hybrid retrieval-summarization architecture. Its itinerary planning engine employs graph optimization under time constraints, delivering personalized travel suggestions. The system emphasizes robustness, linguistic adaptability, and user-centric interactions.

Keywords

Conversational AI, Retrieval-Augmented Generation, Travel Assistant, Intent Classification, Route Optimization

Advisors: Slavko Žitnik

Introduction

Modern travellers seek real-time, tailored information to plan their trips safely and enjoyably. While traditional chatbots operate on static knowledge, Coccodrillo introduces a dynamic, retrieval-augmented architecture capable of integrating current data from web sources and APIs. It assists users with travel decisions involving weather, safety, events, restaurants, attractions, and logistics. The system blends BERT-based intent detection, entity resolution, summarization, and optimization techniques to construct relevant, context-aware responses.

Methods

Our end-to-end pipeline integrates intent classification, entity extraction, and retrieval of real-time data from heterogeneous sources. The system comprises the following components, witch can be also followed on Figure 1.

Intent Classification

We employ the `all-MiniLM-L6-v2` BERT encoder fine-tuned for multi-class intent detection across eight categories.

- Security updates
- Severe weather alerts
- Recommended places to visit
- Concerts/events
- Best restaurants
- Typical food dishes
- Weather forecast
- Travel information

This lightweight model encodes user queries into embeddings, which are subsequently passed through a softmax classifier to infer the most probable intent.

Entity Extraction and Clarification

To identify locations, dates, and other key entities, we use a BERT-based question-answering module (`bert-large-uncased-whole-word-masking-finetuned-squad`). Absent or ambiguous information (e.g., missing travel dates) triggers a clarification subroutine: the system formulates follow-up questions to the user, ensuring complete query parameters. Spelling and formatting errors are corrected via semantic similarity checks against gazetteer (a geographic database) and date lexicons.

Check for Date, City, and Error Handling

The system verifies that, depending on the predicted intent, the corresponding information regarding dates and cities is provided. If any required information is missing, the system returns an empty list (specific tests have been conducted for this subtask, yielding excellent results). Additionally, there is a

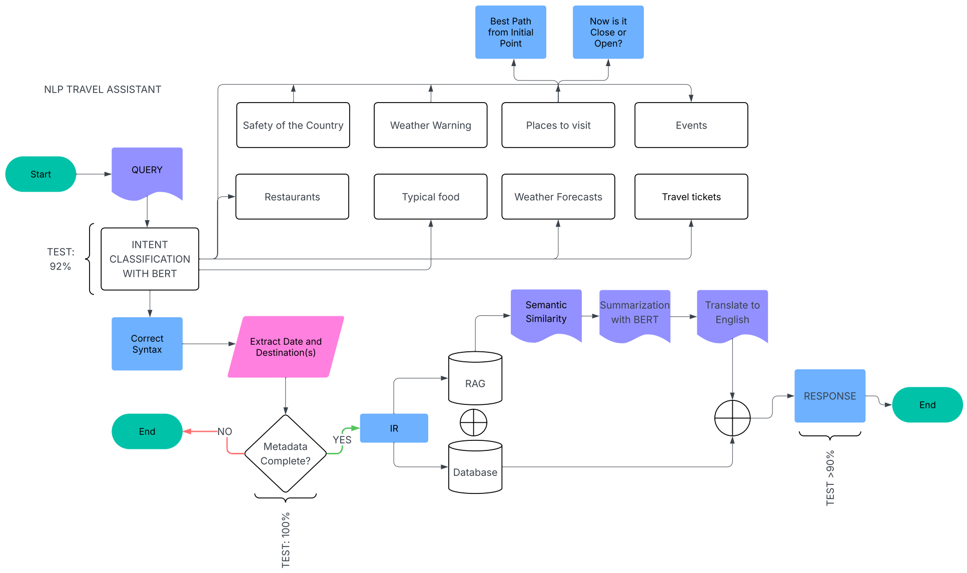

Figure 1. Visualization of the structure of the model.

built-in spell-check feature. It is not necessary to input the city or date (either in a standard date format or in a more casual form, such as "today" or "tomorrow") perfectly; small typos are tolerated, and the system will still be able to understand the input.

Data Retrieval and Integration

Depending on the classified intent, the system retrieves data from different sources. For some intents, such as Recommended places to visit or Typical food dishes, real-time data is not required, as the information is stable over time. In these cases, the system uses a pre-downloaded database, which significantly improves response speed and efficiency.
For other intents that require up-to-date information, such as current events or weather forecasts, the system dynamically retrieves relevant external documents from the web.
The system consults the following data sources:

- **Bing News and Viaggiare Sicuri** for safety and weather alerts in real time. *(on-line)*
- **Lonely Planet and TasteAtlas** (via Selenium & BeautifulSoup) for attractions and local dishes. *(off-line)*
- **Bandsintown** for concerts and events. *(on-line)*
- **Yelp** for restaurant recommendations. *(off-line)*
- **Il Meteo** for extended weather forecasts. *(on-line)*
- **TheTrainLine** for train, flight, and bus schedules. *(on-line)*
- **Google Maps** to verify real-time availability and status of points of interest. *(on-line)*

System Output

The response provide by the system is divided into two parts. The first response is an introductory sentence generated with a **bi-grams model** based on a simple dataset, explaining that the system is searching for information. The second response is the actual structured output of all the information found. The format of the answers depends on the intent identified. For examples of numerous output please see the Appendix B.

System Execution

A detailed overview of how the system must be run in order to function properly is provided in Appendix A.

Strengths and weaknesses of the system

Primary strengths of the system are to automatically correct minor input errors and support queries involving multiple cities and dates within a single sentence.

Furthermore, the system retrieves local news to ensure information is **accurate** and **up to date**. These are then **summarized and translated** into English using the pretrained `facebook/bart-large-cnn` model.

- For **long input documents** the model is not able to translate the document back to English (viz. Appendix A).

To enhance the relevance of retrieved information, the system implements a **semantic similarity ranking** strategy.

- Articles are **compared to the user query** based on content similarity.

- **Only** documents that exceed a predefined similarity threshold are retained.
- This ensures that the final output contains **only the most relevant** and meaningful information.

A Notable Feature of the System

Personalized Sightseeing Recommendations:
When the *Place to visit* intent is classified, the system generates personalized sightseeing recommendations. It constructs a graph where:

- **Nodes:** Represent points of interest, such as museums, monuments, and public squares, with a score reflecting their beauty and an estimated visit duration.
- **Edges:** Denote walking distances between these locations, ensuring an efficient path between them.

See figure 2 for an example of a path in Rome.

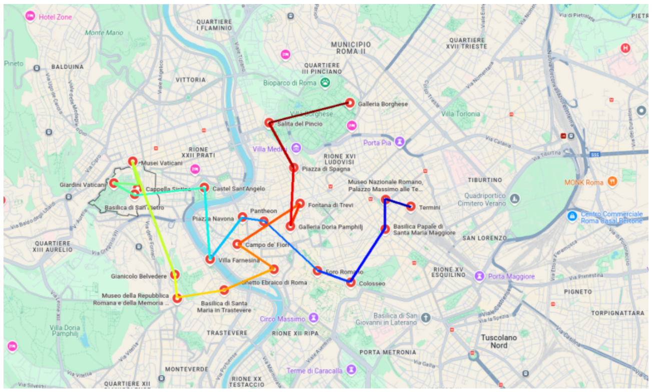

Figure 2. Example of a path in Rome. The colour of the edges represents the order in which the nodes are visited. The route begins with the blue edges and ends with the red ones.

Route Optimization:
Based on this representation, the system applies a route optimization procedure to balance the scenic appeal of locations, walking distances, and total available sightseeing time. Locations that are temporarily closed or inaccessible during the selected travel period are excluded from consideration based on real-time availability data retrieved from Google Maps. The algorithm assumes an average of eight hours per day for sightseeing, and adapts the itinerary accordingly.

Temporary Opening/Closing Check:
The system, using Google Maps, monitors any temporary closures due to ongoing construction, events, or other reasons, and notifies the user if any of the selected destinations are temporarily inaccessible.

Current Supported Cities:
Currently, this optimization module is available for the following cities: Rome, Ljubljana, Prague, Vienna, Florence, Naples, Maribor, Paris, Valencia, Barcelona, and Madrid.

Future Expansion

We plan to extend the list of supported cities in the future. The dataset for additional cities has already been downloaded but requires manual formatting to meet the model's specifications.

Testing and Performance Evalutation

We evaluated system performance using three complementary test suites:

- **Final-output testing**, to verify that correctly posed queries yield appropriate, informative responses (viz. Appendix B, and Final Testing).
- **Intent classification testing**, to measure how often the model assigns the correct intent label.
- **Query-error testing**, to ensure the system gracefully handles under-specified or malformed queries.

Table 1. Test about Intent Classification(show only errors)

True Intent	Predicted Intent	Number of Mistakes
Travel info	Place to visit	1
Security updates	Weather alerts	3
Weather alerts	Future forecasts	1
Typical food	Place to visit	1
Travel info	Future forecasts	1
Security updates	Place to visit	1
Security updates	Travel info	1
Typical food	Best restaurants	1

Test set size: 118 **Accuracy:** 91.53%

Intent Classification testing
Our test set comprised **118 examples** spanning all eight intent categories.
The most frequent confusion occurred between *Safety updates* and *Severe weather alerts*, likely due to overlapping terminology.
Contrary to expectations, *Best restaurants* and *Typical food dishes* exhibited minimal interchange errors.

Final testing
Read System Evaluation and Observations.

Query-error testing
Validated the system's ability to detect missing parameters (e.g., unspecified location or date) and prompt the user for clarification rather than producing nonsensical results.

Limitations

Although our system performs robustly in most scenarios, we identified the following areas for improvement:

- *Rate limiting and blocking*: Automated scraping of transportation schedules (e.g., TheTrainLine) can trigger anti-bot defenses after multiple rapid requests.
- *Translation constraints*: To maximize relevance, safety alerts are fetched in the local language and translated back to English; however, our current pipeline only handles short snippets, necessitating sentence-by-sentence translation.

Technologies Used & Fine-Tuned Model

In this project, various pre-trained models and techniques were adopted to address specific tasks within the field of NLP. Below is a description of the models and configurations used:

- **Intent Detection:**
  The model `all-MiniLM-L6-v2` was used for intent recognition. This lightweight and efficient model generates semantic sentence embeddings, enabling intent identification through similarity comparisons between vectors.
- **Machine Translation:**
  Google Translate was employed for automatic text translation, allowing for multilingual support without the need to train a dedicated translation model.
- **Question Answering:**
  To extract specific information (e.g., "Where is the person going?" or "When does it happen?"), the model `BERT-large-uncased-whole-word-masking-finetuned-squad` was used.
  This model has already been fine-tuned on the SQuAD dataset and is capable of answering questions based on a given context.
- **Text Summarization:**
  For summarization tasks, the model `facebook/bart-large-cnn` was used.
  This BART model is fine-tuned to generate summaries of long texts. It was used in combination with the corresponding `BartTokenizer`.

Comparison with Domain-Specific Fine-Tuned Models

All the models used are pre-trained on general-purpose or standard datasets (e.g., SQuAD for BERT, CNN/DailyMail for BART), and have not been further fine-tuned on our specific dataset.

A potential improvement for future work involves fine-tuning these models on our domain-specific data. Examples include:

- **Intent Detection:**
  Training a model such as BERT or DistilBERT on a custom annotated dataset containing intents relevant to our use case (e.g., travel bookings, specific user requests) could improve intent classification.
- **Custom QA:**
  Fine-tuning the BERT model on a tailored question/answer dataset derived from our application context could enhance the accuracy of contextual answers.
- **Adaptive Summarization:**
  Fine-tuning BART on user-generated texts or short articles similar to our data would likely improve summary quality.

This comparison highlights how the use of pre-trained, general-purpose models provides a solid foundation, but domain-specific fine-tuning can lead to significant performance improvements.

Discussion

The project has achieved satisfactory results: when the system correctly identifies the user's intent, the output is generally reliable. However, it sometimes struggles to accurately extract the specified time frame or date — an aspect that can certainly be improved.

The most significant limitation lies in the intent analysis module, which assumes that every query must fall into one of the predefined intent categories. A useful extension would be to introduce a confidence threshold, below which the system could explicitly respond that it does not handle the given request. This would also require a more robust model capable of interpreting more complex or less structured queries.

Regarding the point-of-interest search algorithm, a transition from a strictly greedy approach to a beam search strategy could prove beneficial, allowing the system to retain multiple promising candidates during route planning.

On the other hand, the implementation of checks for open or closed venues — for instance due to construction or special closures — was particularly effective and valuable for tourists. The same applies to the analysis of local safety conditions, such as extreme weather, natural disasters, or other threats.

Overall, we believe the project represents solid work, with some limited but meaningful opportunities for refinement.

References

For a more detailed explanation, please refer to the README file available on GitHub at the following link.

Appendix A: How to Run the Project

1. Install Dependencies.

To install all required Python libraries and download the necessary BERT models and the `spaCy` language model, execute the following command in the project root directory:

```
python3 ./setup/to_install.py
```

2. Install ChromeDriver.

This project relies on browser automation using Google Chrome, which requires the installation of ChromeDriver. The appropriate version of ChromeDriver for your operating system (Windows, macOS, or Linux) can be downloaded from the official repository from here. **Note:** Precompiled ChromeDriver executables for Linux and Windows are already included in the `./setup/driver` directory for convenience.

Requirements:

- Ensure that the ChromeDriver version matches the installed version of Google Chrome.
- If Google Chrome is not installed, it can be downloaded from here.

Platform-specific instructions:

- **Linux:** Place the `chromedriver` executable inside the `bin/` directory.
- **Windows:** Place the entire `chromedriver_win32` directory in the root directory of the project. Ensure that the executable file is named `chromedriver.exe`.

3. Run the Project.
To execute the project, navigate to the `src` directory and run the main script using the following commands:

```
cd src
python3 run.py
```

Appendix B: Examples

Recommended places to visit

User: *I would like to go to Rome for 3 days. Can you recommend the best things to visit? Start from Termini Station.*

Output:
Greetings, delivering you information about places to visit you asked for recently.

Here's the itinerary for Rome:
Total estimated time: 23 hours

Places to visit:
✓ Palazzo Massimo alle Terme
✓ Basilica di Santa Maria Maggiore
✓ Colosseum
✓ Roman Forum
✓ Pantheon
✓ Piazza Navona
✓ Villa Farnesina
✓ Castel Sant'Angelo
✓ St Peter's Basilica
✓ Vatican Gardens
✗ Sistine Chapel (temporarily closed)
✓ Vatican Museums
✓ Gianicolo
...

Weather alerts and extreme weather conditions

User: *Tell me the latest news about the wether warnings in Valencia.*

Output: News for Valencia:

Date and Time: 03.05.2025 14:00
Description:... live time the Emergency Coordination Center (CEE) has activated this Saturday the alert yellow level by storms inside North Castellón, and for rains and storms throughout the south of Valencia and the entire interior of Alicante. ...

===

Date and Time: 04/22/2025 at 11: 39h.
Description: ... Valencia begins the Easter Week with rain and temperature decrease.The Emergency Coordination Center of the Generalitat Valenciana has established the yellow level alert ...

Safety updates for a location

User: *Hello, what is the current security situation in France, is it safe?*

Output: Hi, here's the information about security situation you were looking for recently.

Posted on 05/03/2014 at 9:00 am GMT.
Intelligence Online exclusively tells the small and large stories of the intelligence world ...

Recommendations on Typical Dishes or Foods

User: *What are the typical food in Naples? And in Paris?*

Output:
Hello, here is the full information you asked for recently regarding typical foods in Naples and Paris.

Top Food Items in Naples

★ **Pizza Napoletana** *(pizza)*
Italy's most emblematic culinary creation, the genuine pizza Napoletana is made with just a few simple ingredients and prepared ...
☆ **Review:** 4.8/5.0

★ **Casatiello** *(Bread)*
A traditional Easter bread from Naples, filled with cured meats and cheese, shaped in a ring and topped with hard-boiled eggs. Symbolically tied to themes of rebirth, it's a staple during the Easter season.
☆ **Review:** 3.9/5.0

★ **Struffoli** *(sweet-pastry)*
Struffoli is a Neapolitan dessert consisting of small, deep-fried balls of dough that are soaked in honey. Traditionally, struffoli is prepared at Christmastime ...
☆ **Review:** 3.6/5.0

Top Food Items in Paris

★ **Baguette** *(Breads)*
Probably the most iconic image of France, this long, light, airy bread with its characteristically crispy crust dates back to the 19th century, when wheat and white breads were no longer a privilege of the rich ...
☆ **Review:** 4.4/5.0

★ **Macarons** *(Cookies)*
These small, round, sweet meringue-based cookie sandwiches with filling in the middle are light and crunchy on the outside and chewy in the middle ...
☆ **Review:** 4.0/5.0

★ **Croque-monsieur** *(Sandwich)*
A hot sandwich made with ham and Gruyère cheese, sometimes topped with Mornay sauce. Popularized in early 20th-century Paris and famously mentioned by Marcel Proust.
☆ **Review:** 4.2/5.0

1. System Evaluation and Observations

We conducted tests on the system using 30 queries covering the 8 different intents, varying the difficulty and ambiguity levels of the questions. Below, we provide our evaluation and observations.

The travel-focused RAG chatbot **Coccodrillo** demonstrates solid foundational capabilities in handling a wide array of user intents related to trip planning and real-time travel information. These intents include destination recommendations, transportation guidance, cultural and culinary tips, weather and safety alerts, and event listings. The system exhibits commendable strengths, particularly in formatting, geographic breadth, and structured information delivery. However, there are consistent issues related to personalization, content clarity, data completeness, and linguistic polish.

Key Strengths

1. **Content Coverage and Comprehensiveness**
   The chatbot handles diverse user intents well, from practical concerns like severe weather alerts and public transport logistics to more experiential aspects such as food recommendations and concerts. Its answers tend to be informative, with consistent formatting (e.g., use of tables, bullet points, emojis), making content digestible. For example, responses about *"Recommended Places to Visit"* are well-structured and include cultural landmarks tailored by trip duration.
   Additionally, it performs **real-time online checks to verify whether places are open or closed**, which is crucial information for tourists.

2. **Geographic and Temporal Adaptability with Advanced Filtering**
   For intents such as weather forecasts, event calendars, and restaurant recommendations, the system accurately adapts content to the requested city and time frame. For example, in event searches, the system dynamically includes nearby cities when necessary, and food recommendations are linked to local traditions and ingredients.
   Additionally, it is possible to filter events by singer, search for multiple singers simultaneously, and combine multiple cities and time periods within the same query. These last functionalities are available for any geographic location and for any type of intent, not just musical events; for instance, it is possible to search multiple cities or time frames for other types of queries as well.

3. **Structured and Friendly UX**
   The chatbot's outputs frequently use visually appealing formatting — like clear headers, dividers, and symbols — to support readability. This is notable in categories like *"Typical Food Recommendations"* and *"Best Restaurants"*, where list structure and stylistic elements help user engagement.

4. **Scalability and Robustness**
   The assistant is scalable and robust because it can detect when requested information is missing. For online searches, it works across all cities worldwide, ensuring broad geographic coverage. Additionally, the system incorporates automatic correction for typos related to city names and time periods, making the analysis more robust and tolerant of user input errors.
   It is possible to perform simultaneous searches across multiple cities and time periods for the same intent, enhancing query flexibility.
   From an extensibility standpoint, the system is designed to easily integrate more interactive and personalized features in the future. For example, if a user wants to explore a city and asks, "Show me things to see, I am an art lover," the system—which already has all available attractions stored in its database—can rank places to visit based on an algorithm considering parameters such as beauty, visit duration, and distance between locations. The output thus provides an optimized itinerary according to these criteria.
   The ranking algorithm is modular: it currently uses three parameters, each assigned a weight percentage that determines its influence on the final order. To extend the system, new parameters can be added with custom priority weights. For instance, features can be assigned to each place (e.g., art, sports, nature, etc.) and incorporated into the ranking calculation, making personalization simple and scalable.

Areas for Improvement

1. **News Update**
   The system returns the most recent news but does not search for a specific date (which is often difficult to find within the article itself). Consequently, if the most recent news is still outdated, the system will return it anyway. There is a similarity check between the documents and the initial query to ensure that the returned documents are relevant to the search.

2. **Interactivity and Response Design**
   The chatbot is capable of recognizing vague or incomplete queries, as demonstrated by a specific test called query error testing. However, in such cases it simply responds with an empty string, without any further interaction. There is no ongoing user-chatbot dialogue where the system asks, for example, “Could you repeat that? I didn’t understand,” to clarify missing information. This lack of interactivity reduces the overall effectiveness of the conversation. Improvements in dynamic response handling and fallback mechanisms are needed to enable more robust and effective dialogue.