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 travelers 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, namely: Security updates, Severe weather alerts, Recommended places to visit, Concerts/events, Best restaurants, Typical food dishes, Weather forecast and 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 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.

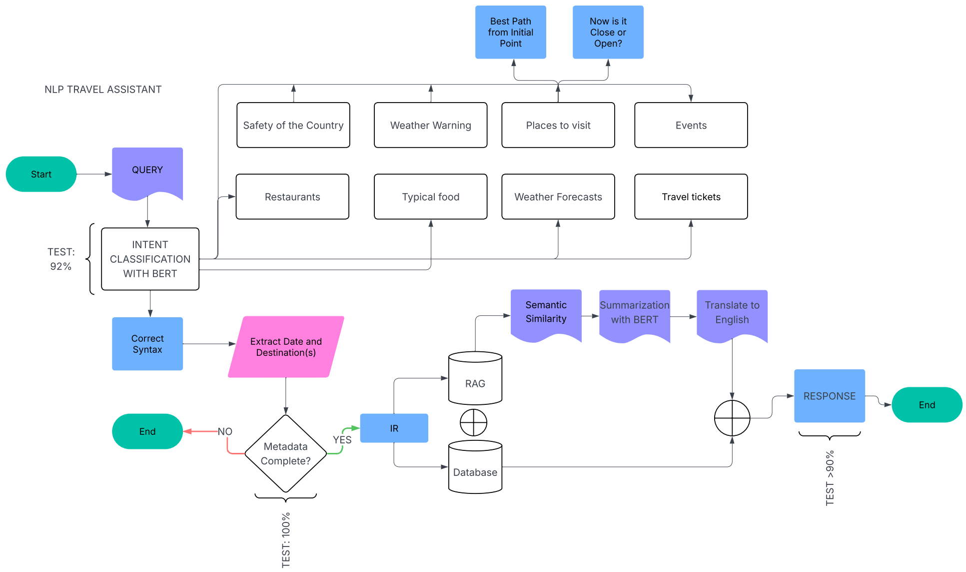

Figure 1. Visualization of the structure of the model.

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, for safety-related queries concerning foreign locations, the system retrieves relevant news in the local language to ensure the information is accurate and up to date. These texts are subsequently summarized and translated into English using pretrained `facebook/bart-large-cnn` pipeline. However this approach is also a weakness, because for long input documents the model in not able to translate the document back to English (viz. Appendix A).

To enhance the relevance of retrieved documents, the system ranks articles according to their similarity to the user query. Only those texts that surpass a predefined similarity threshold are retained and included in the final output.

A notable feature appears when the *Place to visit* intent is classified. The system is able to generate personalized sightseeing recommendations. To achieve this, it constructs a graph where nodes represent points of interest—such as museums, monuments, and public squares—and edges denote walking distances between them. See figure 2 for a example of a path in Rome. Each node is further characterized by two attributes:

a score reflecting the aesthetic or cultural significance of the site, and an estimated visit duration.

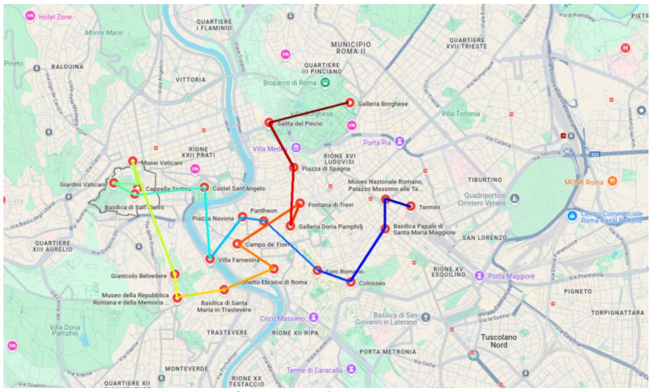

Figure 2. Example of a path in Rome. The colour of the edges represents the order in which the nodes are visited. The route begins with the blue edges and ends with the red ones.

Based on this representation, the system applies a route optimization procedure designed 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.

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.

System Execution

Users invoke the system via `run.py`, which orchestrates the above modules, compiles retrieved data, and formats the final response into a coherent, structured summary.

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.
- **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.

In the final-output tests, we submitted well-formed questions and confirmed that the agent both recognized the intent and returned coherent, contextually relevant information.

For intent classification, our test set comprised **118 examples** spanning all eight intent categories.

The model achieved **91.53% accuracy**. Misclassifications are visualized in Fig.3. 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. Finally, 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.

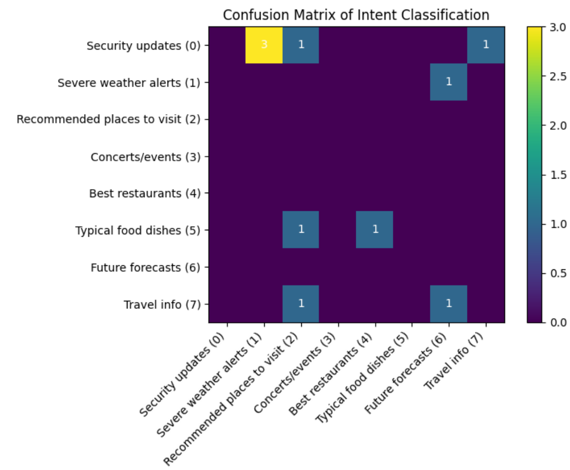

Figure 3. Intent Classification Confusion Matrix. Rows denote the true intent; columns denote the predicted intent.

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.

Future Work

Prior to final submission, we will:

- Implement request throttling and caching to mitigate blocking by external sites.
- Integrate fallback data sources to improve robustness when primary website fail.
- Extend the translation module to batch-process longer texts efficiently.

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.

Acknowledgments

Here you can thank other persons (advisors, colleagues ...) that contributed to the successful completion of your project.

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: https://sites.google.com/chromium.org/driver/ **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: https://www.google.com/chrome/

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

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.

France:

——————————————-

Publié le 24/02/2021 à 3h30 GMT Tous les matins, à 4h GMT Depuis 40 ans, Intelligence Online raconte en exclusivité les petites et grandes histoires du monde du renseignement, de ceux pour qui l'information est un instrument de travail, et fait la lumière sur leurs réseaux. ...

Posted on 05/03/2014 at 9:00 am GMT Every morning, at 4 hours GMT for 40 years, Intelligence Online exclusively tells the small and large stories of the intelligence world, of those for whom information is a working instrument, and sheds light on their networks. ...

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 for rains and storms throughout the province of Castellón and the north of Valencia. ...

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 wou asked for recently.

Here's the itinerary for Rome:

Total estimated time: 23 hours

Places to visit: ✓ Museo Nazionale Romano: Palazzo Massimo alle T

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

Output:

subsection*Best restaurants to eat **User:** *Can you write for me the best places where i can eat in Prague.*

Output:

subsection*Recommendations on typical dishes or foods **User:** *What are the typical food in Naples? and in Paris?*

Output:

subsection*Best restaurants to eat **User:** *I am going in Berlin, tell me the temperature for friday.*

Output: