Conversational Assistant

Kyriakos Schwarz - 17.10.2024

Agenda

1. Challenge and Goals
2. Solution Steps
3. Input data preprocessing
4. Data Analysis
5. Testing strategy
6. Training results
7. System Architecture
8. Business logic
9. Demo
10. Improvement Ideas

Challenge and Goals

- A user is interacting with a conversational assistant
  - E.g., to find and book an appointment
- The conversational assistant should understand **what the user means**
  - The “intent” that represents the meaning of the user input
  - E.g., “i want to book an appointment” → “book_appointment”

- **Goals**:
  - Design and implement a web interface for intent classification
  - take input sentence, display intent, save in db, display history of predictions

Solution Steps

- Understanding and preprocessing of the data
- Developing a train-test strategy and evaluating the results
- Generating model artefacts for inference use
- Develop an inference module
- Developing a web interface with the required elements
- Setting up a database, authentication, tables, db module
- Developing a flexible module architecture for connecting the modules in a decoupled fashion (separation of concerns)
- Testing and improving the whole app
- Ensuring clearly structured and documented code
- Presenting the results

Disclaimer

- Those are a lot of tasks for a very short amount of time
- My time is very limited (full time job & long commute)
- Each step would require a much larger dedicated amount of time
- Some steps are in practice fairly new to me

Input Data

Input Data: 197 rows of two columns: intent, text

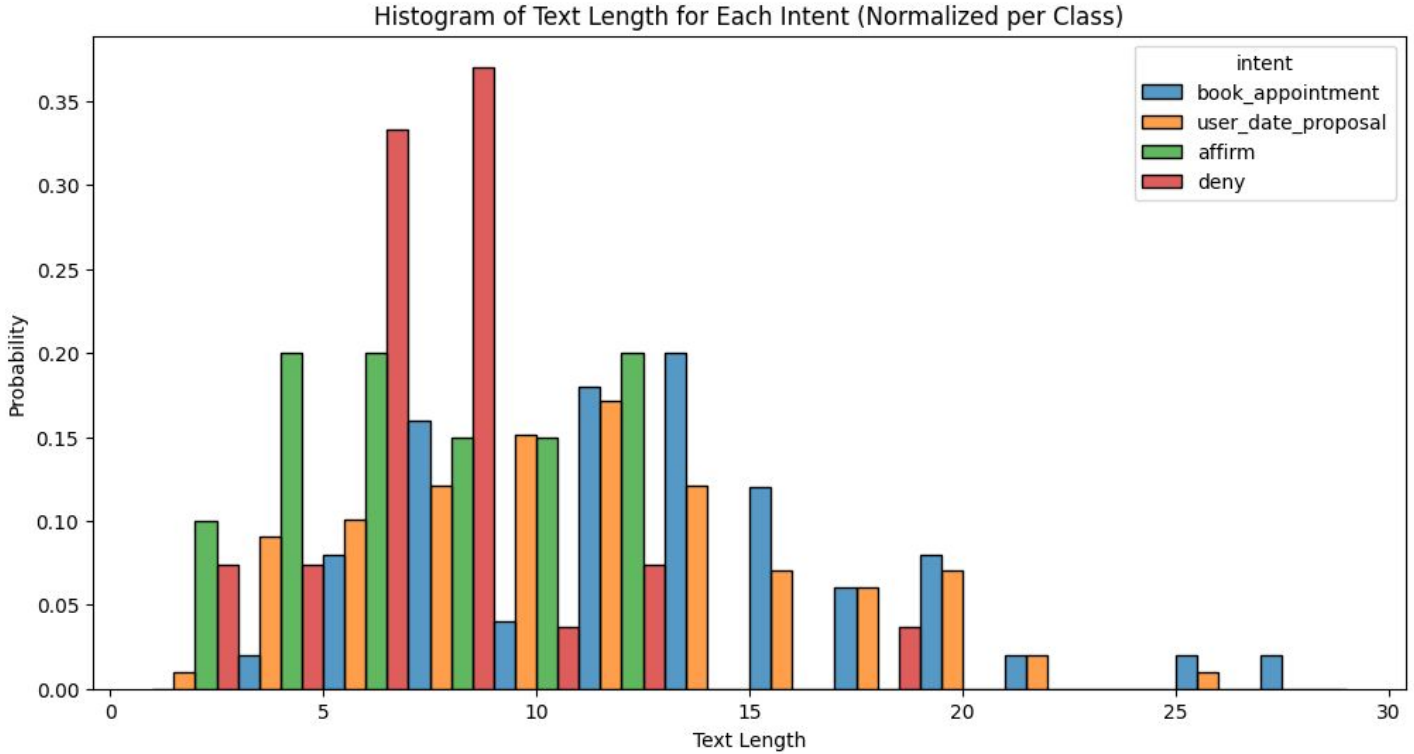
Intent	Count
user_date_proposal	100
book_appointment	50
deny	27
affirm	20

→ Very small dataset, **imbalanced classes**

Data Preprocessing

- Convert to lowercase
- Fix contractions (e.g., “can’t → cannot”, “that’s → that is”)
- (Stopword removal) → Did not work well with the meaning of text (remove “**not**”)
- Remove punctuation and other non-alphanumeric characters
- Word lemmatization: (e.g., “rocks → rock”, “better → good”)
- Remove duplicates

Data Analysis



Data Analysis

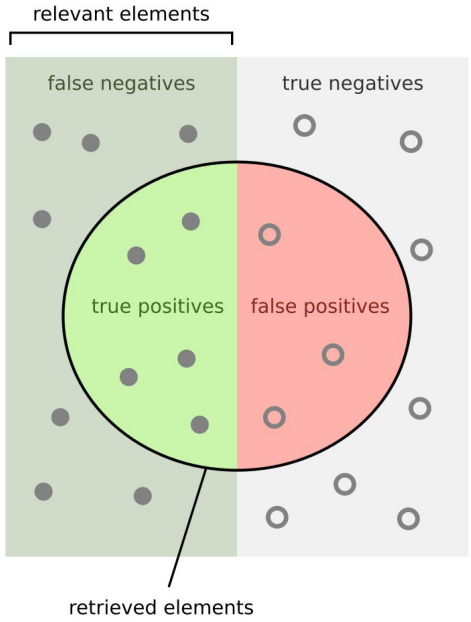
- Identification of unique words per class:
  - book_appointment: appointment, car, would, need, like, service, schedule
  - user_date_proposal: afternoon, wednesday, about, pm
  - affirm: yes, perfect, monday, okay, nine, thanks, thank
  - deny: no, sorry, available, so, early, other, option

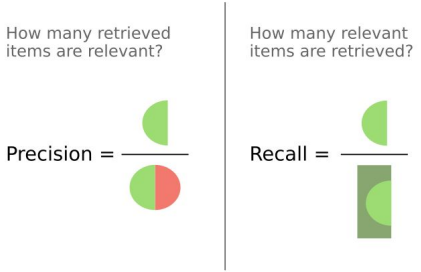
- Suitable data for a Tfidf Vectorizer

Testing

- Train-test split with stratify on “intent”, 20% test data
- 5-fold cross validation
- Evaluation on F1-score
  - harmonic mean of:
    - Precision (fraction of relevant instances)
    - Recall (fraction of relevant instances that were retreived)
  - useful for imbalanced datasets
  - may overemphasize performance on minority classes

- Results: 0.82, 0.60, 0.71, 0.77, 0.77





Source: https://en.wikipedia.org/wiki/Precision_and_recall#/media/File:Precisionrecall.svg

Training

- Select the model with the best F1-score (Fold 1: 0.82)
- Train on the whole train set
- Results:

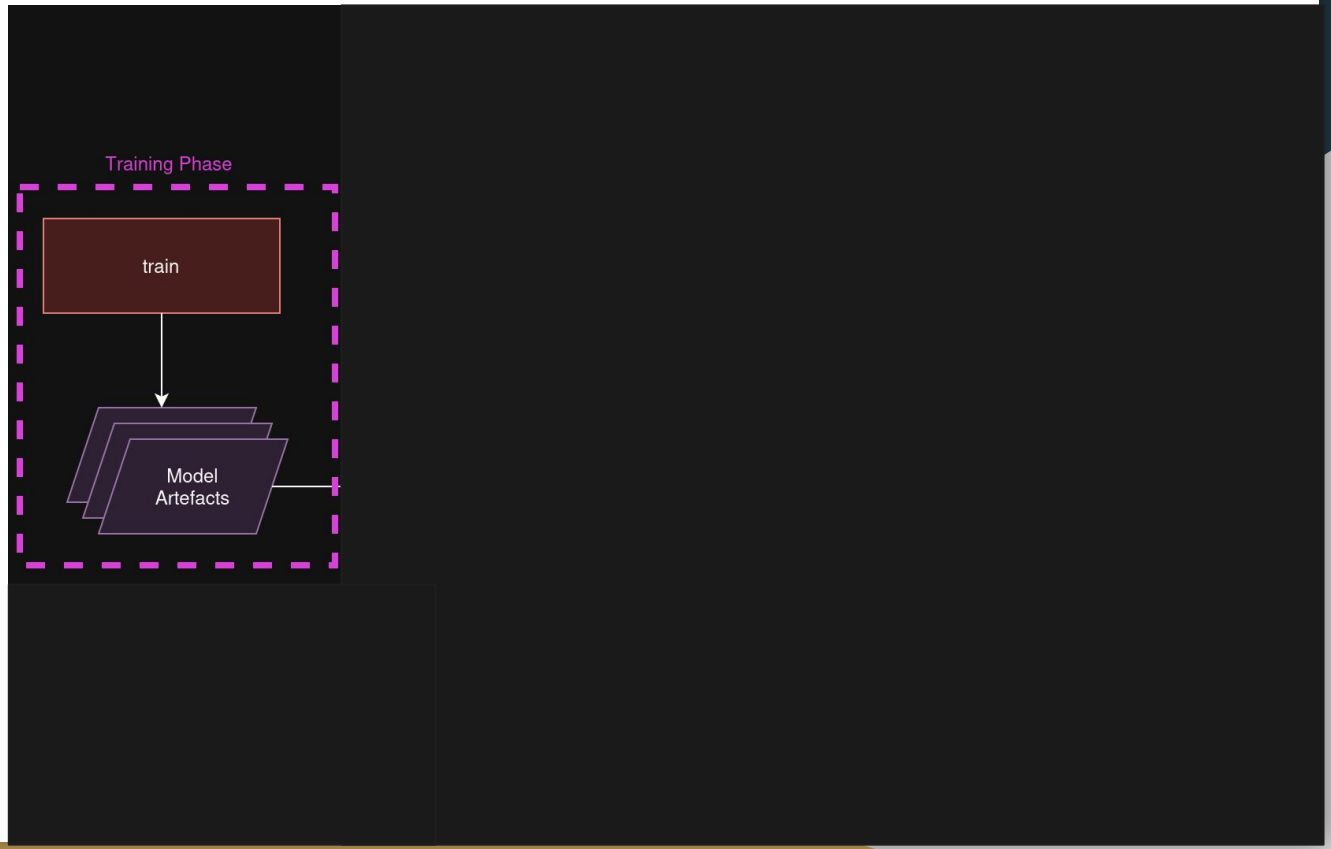
Classification Report with Original Class Names:

	precision	recall	f1-score	support
affirm	0.75	0.75	0.75	4
book_appointment	1.00	1.00	1.00	10
deny	1.00	1.00	1.00	6
user_date_proposal	0.95	0.95	0.95	20
accuracy			0.95	40
macro avg	0.93	0.93	0.93	40
weighted avg	0.95	0.95	0.95	40

- Produced artefacts:
  - model, tfidf, label encoder

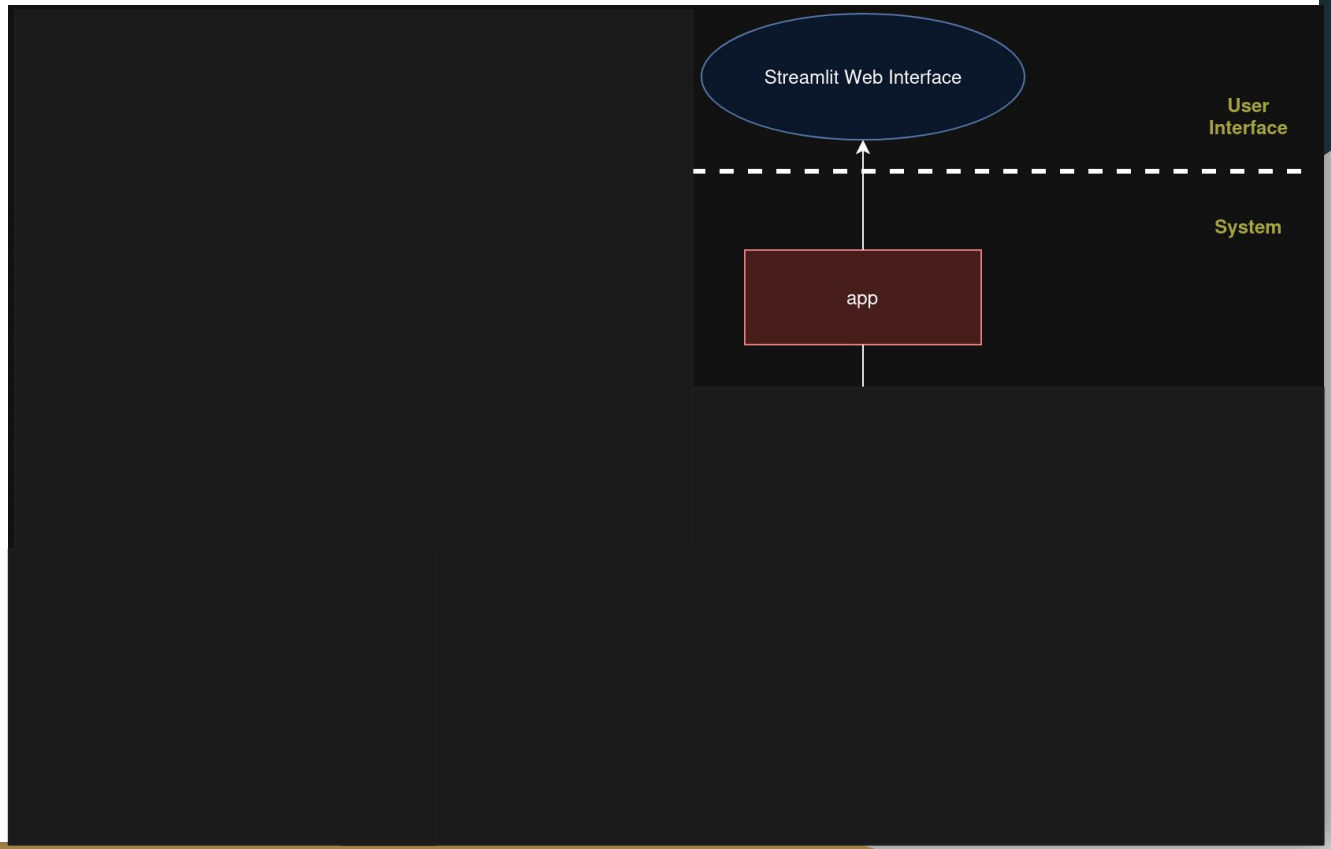
System Architecture

- A **train** module generates the Model Artefacts:
- model, tfidf, label encoder



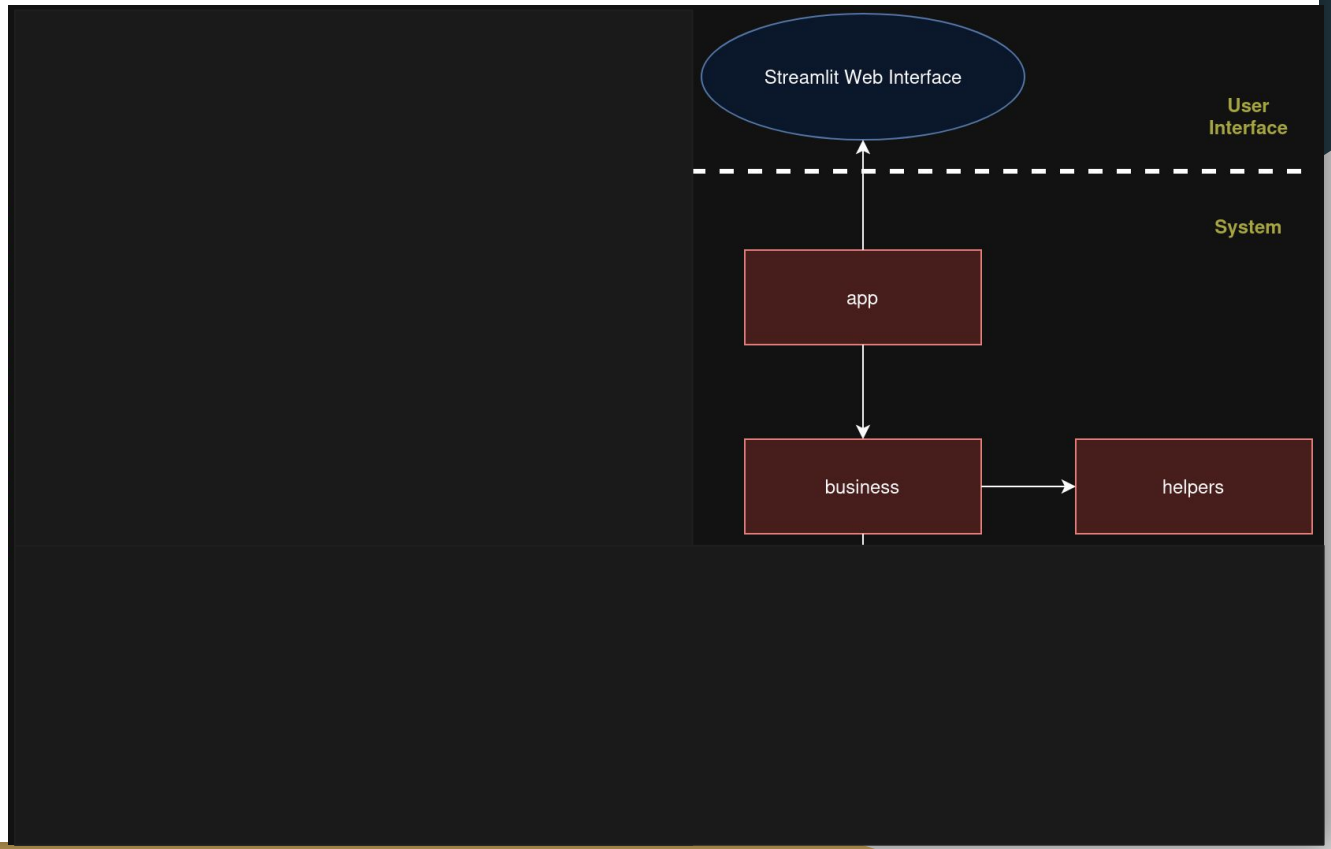
System Architecture

- A Python-based Webapp framework (**Streamlit**) allows for fast prototyping
- **app** module only contains frontend elements (text input, buttons, etc.)
- also functions as entry point of the application



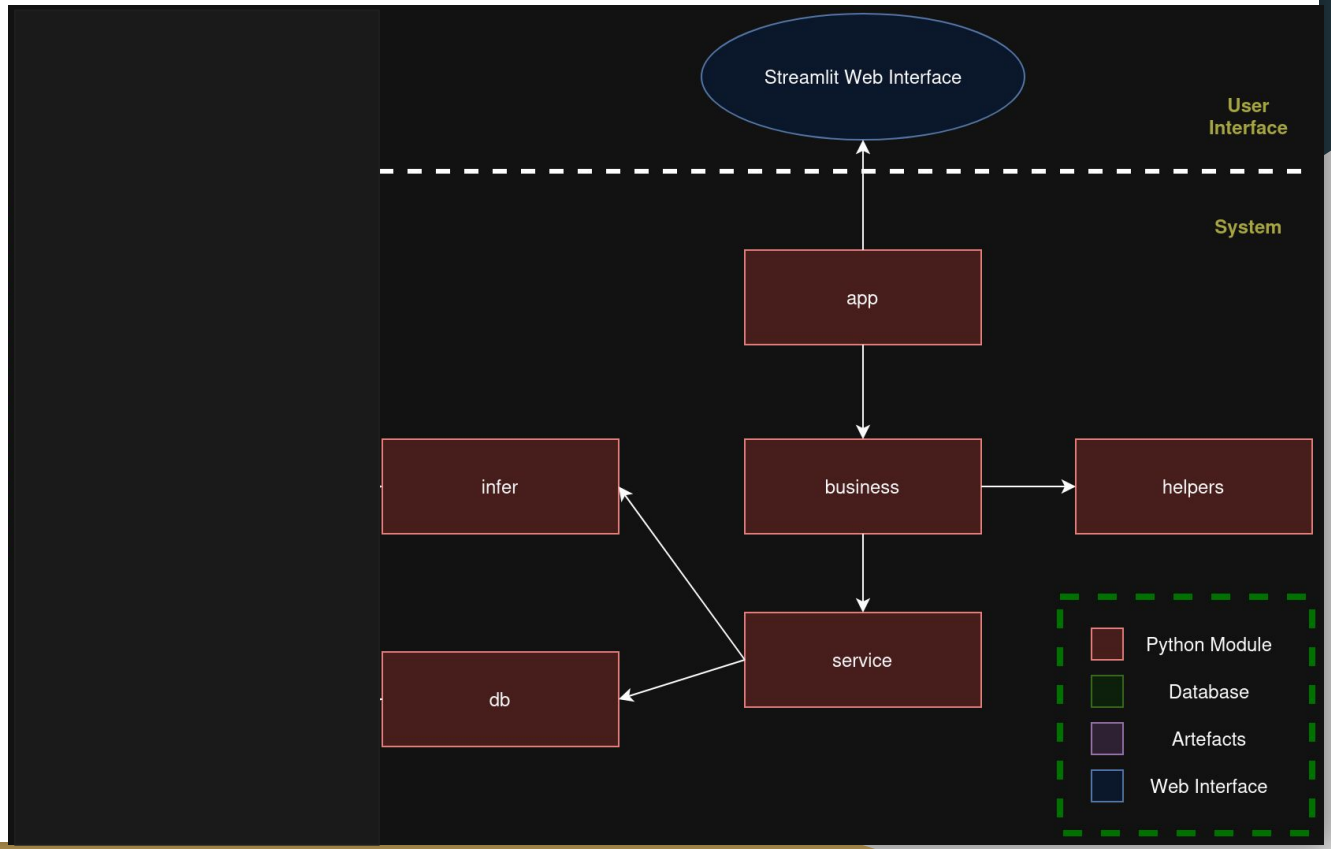
System Architecture

- A **business** module determines what happens when the user interacts with the **app** elements
- Additional **helper** functions take care of input text cleaning, formatting, etc.



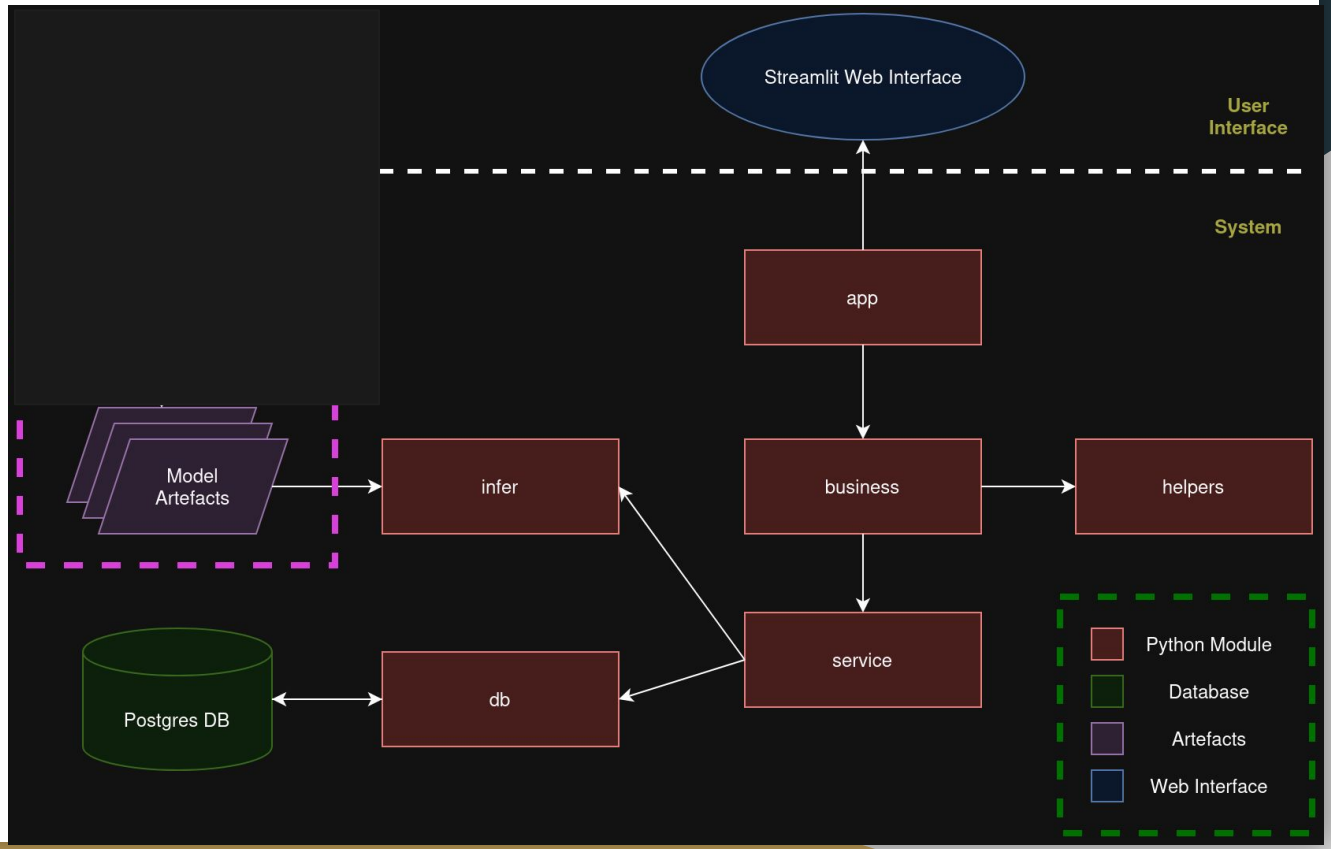
System Architecture

- A service layer provides an abstraction over components which can be replaced (e.g., different Database), acts like an **interface**
- Enables decoupling and provides a **flexible system** of independent components



System Architecture

- An **infer** module utilizes the trained **Model Artefacts** for predictions on **new text** input
- A **db** module provides the appropriate connections and queries to a specific database flavor (**Postgresql**)

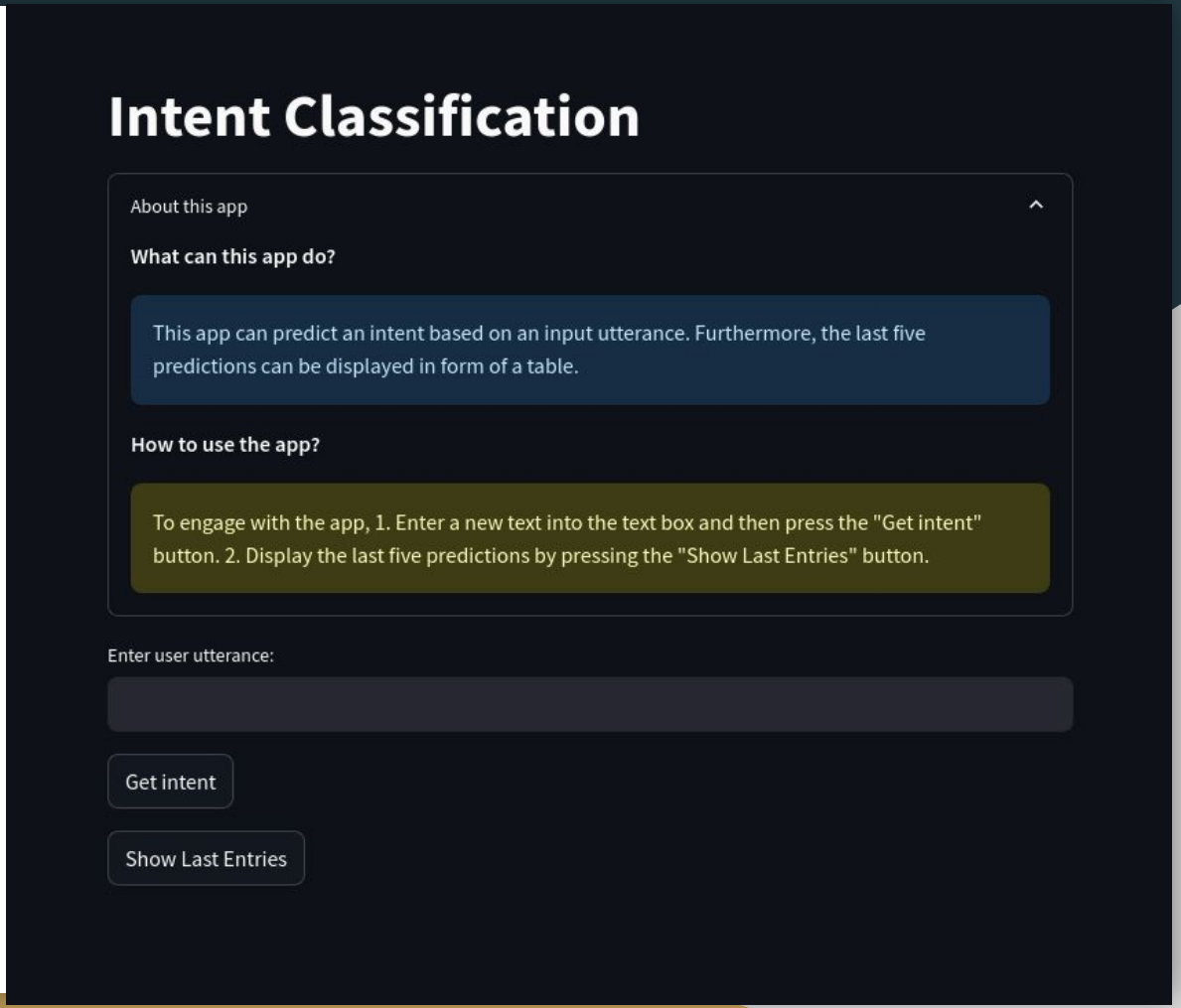


Business Logic

- Assumption: A confident prediction is needed to determine a class
- If max probability > 0.5 display class
- Otherwise “Intent unclear”

- Display the last five entries in the DB (chronologically)

Demo Time!


Intent Classification
About this app
What can this app do?
This app can predict an intent based on an input utterance. Furthermore, the last five predictions can be displayed in form of a table.
How to use the app?
To engage with the app, 1. Enter a new text into the text box and then press the "Get intent" button. 2. Display the last five predictions by pressing the "Show Last Entries" button.
Enter user utterance:
Get intent
Show Last Entries

Improvement Ideas

- Small dataset → Generate more samples per class
- Possibly add a fifth “unclear” class with irrelevant text
- Implement stopword removal without changing of meaning
- Add more features to the training data (e.g., length of text, nr of words)
- Handle special words (e.g., weekdays)
- Try different models besides LR
- Abstract the Frontend implementation for easy replacement
- Improve UI/UX
- Containerize the App

Thank you!