🧠 Purpose of Training

The chatbot is designed to **assist university users** by handling queries related to:

- Greetings and farewells
- Library opening hours
- Campus locations (e.g., library, cafeteria)
- Daily cafeteria menu
- Professors' office hours

📁 Training Data Used

✅ nlu.yml

Contains user input examples for 6 intents:

- greet, goodbye, ask_hours, ask_location, ask_menu, ask_schedule
- Uses labeled entities like entity (e.g., library) and professor (e.g., Dr. Jones)

✅ domain.yml

Defines:

- **Intents** and **entities**
- Bot **responses** (e.g., utter_library_hours)
- Custom **actions**:
  - action_retrieve_location
  - action_retrieve_menu
  - action_retrieve_schedule

✅ stories.yml

Defines **conversation flows** for:

- Responding to each intent with either a **static message** or a **custom action**
- Handling fallback (unrecognized intent)

🚀 Execution Command

!source /usr/local/rasa_venv/bin/activate && rasa train

This command **activates the virtual environment** and trains the Rasa model on the above files.

🎯 Rasa Training Process

The **training process in Rasa** serves to build **two core models** that power the chatbot's behavior:

1. **Natural Language Understanding (NLU) Model**
    - **Goal**: Understand user messages by extracting:
        - **Intent** (e.g., ask_location)
        - **Entities** (e.g., library, Dr. Jones)
    - **Training Data**: Comes from **nlu.yml**.
2. **Dialogue Management (Core) Model**
    - **Goal**: Learn how to respond appropriately based on:
        - Current user intent
        - Past conversation history
    - **Training Data**: Comes from **stories.yml** and **rules.yml**.

🧠 Models Used in Training

Based on Rasa 3.1 and default behavior, the models used during the training process are:

🔹 **For NLU:**

- **Tokenizer**: WhitespaceTokenizer
- **Featurizer**: CountVectorsFeaturizer or DIETClassifier
- **Classifier**: DIETClassifier (Dual Intent and Entity Transformer)
    - Handles both intent classification and entity extraction using a **transformer architecture**.

🔹 **For Dialogue Management:**

- **Policy**: RulePolicy
    - Executes predefined rules (e.g., ask_hours → utter_library_hours).

- **Policy**: MemoizationPolicy
    - Remembers previously seen conversations from stories.yml.
- **Optional**: TEDPolicy (Transformer Embedding Dialogue)
    - Learns general conversation patterns not covered by memorization or rules.

These are configured in **config.yml**, which defines the pipeline and policies. If it's not specified, Rasa uses default settings.

📘 Summary

Component	Purpose	Model Used
NLU	Classify intents, extract entities	DIETClassifier
Dialogue	Predict next bot action	RulePolicy, MemoizationPolicy (possibly TEDPolicy)

🎯 Purpose of test_nlu.yml

This file contains **example user messages** along with their **correct intent labels** and **entities**. It is used to check how well the chatbot's NLU model can:

- Detect the correct **intent** (what the user wants)
- Identify the correct **entities** (specific details like names, places, etc.)

🧪 What Happens During rasa test nlu

1. ✅ **Rasa loads the trained model**
   - This model was built using training data from nlu.yml.
2. ✅ **It reads test_nlu.yml**
   - It goes through each example like:

     "where is the library?"

   - It knows that the correct intent is ask_location and the entity is library.
3. ✅ **Model makes predictions**
   - The model guesses what the intent and entities are for each example.
   - Rasa compares these predictions with the correct answers from test_nlu.yml.
4. ✅ **Results are saved and displayed**
   - 📄 Intent classification report: results/intent_report.json
   - 📄 Entity extraction report: results/DIETClassifier_report.json
   - 📊 Confusion matrix: Shows where the model made correct or wrong predictions

✅ Example Use

From test_nlu.yml:

```
- intent: ask_schedule
  examples: |
    - when can I meet [Dr. Jones](professor)?
```

Model's Task:

- Intent → Should predict ask_schedule
- Entity → Should extract Dr. Jones as professor

If the prediction is correct, it contributes to the model's **accuracy score**.

📌 Summary

Step	What it does
Load Model	Use the trained model from previous steps
Read Test File	Use test_nlu.yml to evaluate performance
Compare Predictions	Check predicted vs. expected intents/entities
Save Results	Report accuracy and errors to results/ folder

To analyze your **Rasa NLU test results**, here's how to interpret the output

```
2025-05-22 13:27:51 WARNING  rasa.shared.utils.common  - The UnexpecTED Intent Policy is currently experime
2025-05-22 13:27:51 INFO     rasa.nlu.test  - Running model for predictions:
100% 18/18 [00:00<00:00, 22.21it/s]
2025-05-22 13:27:52 INFO     rasa.nlu.test  - Intent evaluation results:
2025-05-22 13:27:52 INFO     rasa.nlu.test  - Intent Evaluation: Only considering those 18 examples that ha
2025-05-22 13:27:52 INFO     rasa.nlu.test  - Classification report saved to results/intent_report.json.
2025-05-22 13:27:52 INFO     rasa.nlu.test  - Incorrect intent predictions saved to results/intent_errors.j
2025-05-22 13:27:53 INFO     rasa.utils.plotting  - Confusion matrix, without normalization:
[[3 0 0 0 0 0]
 [1 2 0 0 0 0]
 [0 0 3 0 0 0]
 [0 0 0 3 0 0]
 [0 0 0 0 3 0]
 [0 0 0 0 0 3]]
2025-05-22 13:27:55 INFO     rasa.nlu.test  - Entity evaluation results:
2025-05-22 13:27:55 INFO     rasa.nlu.test  - Evaluation for entity extractor: DIETClassifier
2025-05-22 13:27:55 INFO     rasa.nlu.test  - Classification report saved to results/DIETClassifier_report.
2025-05-22 13:27:55 INFO     rasa.nlu.test  - Incorrect entity predictions saved to results/DIETClassifier_
2025-05-22 13:27:55 INFO     rasa.utils.plotting  - Confusion matrix, without normalization:
[[ 3  1  0]
 [ 0 63  0]
 [ 0  0  6]]
/usr/local/rasa_venv/lib/python3.8/site-packages/rasa/utils/plotting.py:284: UserWarning: Attempting to set
  axes[side].set(yticks=yticks, xlim=(0, x_ranges[side]), ylim=y_range)
```

📊 Intent Confusion Matrix

```
[[3 0 0 0 0 0]
 [1 2 0 0 0 0]
 [0 0 3 0 0 0]
 [0 0 0 3 0 0]
 [0 0 0 0 3 0]
 [0 0 0 0 0 3]]
```

This matrix compares **true intents** (rows) with **predicted intents** (columns).

Each row = **true intent class**
Each column = **predicted intent class**

🔍 Example Interpretation:

- **Class 0 (1st row):** 3 correct predictions
- **Class 1 (2nd row):** 2 correct, 1 misclassified as class 0
- **Classes 2 to 5**: 3 correct each

🔴 The only error is:

- **1 instance of intent class 1 was predicted as class 0**

✅ Entity Confusion Matrix

```
[[ 3  1  0]
 [ 0 63  0]
 [ 0  0  6]]
```

Interpretation

Each row = actual (true) entity label
Each column = predicted entity label

Class	Meaning (assumed)	True Samples	Correct	Errors
0	(e.g., entity)	4	3	1 predicted as class 1
1	(e.g., professor)	63	63	0 errors
2	(e.g., location)	6	6	0 errors

Findings

- ✅ **Class 1 and 2**: Perfect entity extraction — **no errors**
- ⚠️ **Class 0**:
  - 3 correct predictions
  - 1 mistake: it was predicted as class 1