Business Analytics : Building an E-commerce Chatbot

Maxime Manderlier

December 8, 2025

Introduction

In this session, you will transform a language model into a simple **e-commerce chatbot** capable of understanding user requests, asking clarification questions, and recommending *real products* from a dataset.

You may use:

- the **fine-tuned model** from the previous TP, or
- a **base LLM** without fine-tuning.

Both options are acceptable. The focus of this TP is on **designing a reasoning pipeline** around the model, not on training it further.

Dataset

We will use a public sample of Amazon products (metadata only):

`https://github.com/luminati-io/Amazon-dataset-samples`

You can load the dataset directly from the web as follows:

```
import pandas as pd
import requests
import io

url = "https://raw.githubusercontent.com/luminati-io/Amazon-dataset-samples/main/amazon-pr
↪  oducts.csv"
r = requests.get(url)

if r.status_code != 200:
    raise RuntimeError(f"Failed to download dataset: {r.status_code}")

df = pd.read_csv(io.StringIO(r.text))
df.head()
```

Each row represents a product with fields **such as** `title`, `price`, `description`, and `category`. Your chatbot will recommend items chosen *directly* from this dataset.

Chatbot Behaviour

Your chatbot must be able to:

- understand the user's request,
- determine whether enough information has been provided,
- ask for clarification when needed,
- select products from the dataset using simple filtering,
- generate clear, helpful recommendations.

Required information

To produce a recommendation, the chatbot must know at least:

- **what kind of item the user is looking for** (e.g., wireless headphones, gaming laptop, running shoes), and
- a **maximum price**.

If any of these elements are missing, the chatbot must ask a clarification question, e.g.:

> *"To help you better, could you tell me what type of product you are looking for and/or your maximum budget?"*

Product Selection Pipeline

A simple selection pipeline is sufficient:

1. Filter products by matching the requested item type in the title or description.
2. Filter by maximum price.
3. Optionally filter by additional keywords extracted from the user's query.
4. Return the top–k products (e.g., 3 or 5).

Your chatbot must not invent products. All recommended items must come from the loaded dataset.

Extracting Information Using the LLM (Chat Template)

You may use the model itself to extract structured information from the user's query. Below is an example using a **chat template** style, where we build a list of messages and let the model return a JSON object.

```python
from transformers import AutoTokenizer, AutoModelForCausalLM, pipeline
import json
import torch

model_name = "./data/models/Llama-3.2-3B-Instruct"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name, device_map="auto")

user_query = "I am looking for wireless headphones under 80 euros."

messages = [
    {
        "role": "system",
        "content": (
            "You are an e-commerce assistant. "
            "Given a user query, extract the following fields and return JSON only:\n"
            "- item_description: a short description of what the user wants\n"
            "- max_price: numeric value if provided, otherwise null\n"
            "- keywords: list of optional keywords\n"
            "- missing_information: list of required fields not provided by the user\n"
            "Required fields: item_description, max_price.\n"
            "Only return the JSON so that it is parsable using json.loads(response). Do not
            ↪  add any other text or information."
        )
    },
    {"role": "user", "content": user_query}
]

# Build the chat completion pipeline
pipe = pipeline(
    "text-generation",
    model=model,
    tokenizer=tokenizer,
    device_map="auto"
)

# Apply chat template
prompt = tokenizer.apply_chat_template(
    messages,
    tokenize=False,
    add_generation_prompt=True
)
```

```python
# Generate only the assistant's message
response = pipe(
    prompt,
    max_new_tokens=200,
    return_full_text=False
)[0]["generated_text"]

print(response)

# Parse JSON
try:
    extraction = json.loads(response)
except Exception:
    extraction = None
```

The returned JSON dictionary (e.g. `item_description`, `max_price`, `keywords`) can then be passed to your product selection function.

Expected Workflow

A typical conversation should follow these steps:

1. The user sends a request.
2. The model extracts structured information (item description, max price, keywords).
3. If required information is missing, the chatbot asks a clarifying question.
4. Otherwise, the system selects real products from the dataset.
5. The model generates a recommendation.

Going Further (Optional)

A few simple extensions can be explored to enhance the assistant:

- Ask more detailed follow-up questions (e.g., preferred style, brand, or additional constraints).
- Introduce a small ranking strategy, for instance favouring items with more keyword matches or a lower price.
- Allow multi-turn refinement such as “Show me cheaper options” or “Only wireless models”.
- Provide short product descriptions to the LLM so it can justify recommendations using real item features.