Business Analytics – TP5

Maxime Manderlier
Fabian Lecron

Faculté Polytechnique
Université de Mons

December 1, 2025

Recap TP3–TP4: Churn Prediction

- Business task: predict whether a customer will **churn** or **stay**.
- We fine-tuned a **LLM** to output `CHURN` vs `NO_CHURN`.
- We evaluated the model using standard metrics:
  - precision, recall, F1-score, accuracy, . . .
- Using a large LLM for this task may be **unnecessarily complex and costly** compared to simpler alternatives.

TP4: Towards Explanations

- Goal: not only **predict** churn, but also **explain why**.
- Using **interpretable models**, such as:
  - decision trees
  - logistic regression
  - ...
- Their performance was **comparable** to the LLM.
- Idea: use the model's decision logic to produce **transparent explanations**.

Explanations + LLMs: What We Could Do

- Ideally: build a **dataset of explanations** (label + clear justification).
- Then fine-tune an LLM to output:
  - both the **prediction** and the **explanation**.
- Limitation: interpretable models from TP4 have **moderate performance**.
  - wrong prediction → misleading explanation.
- In practice: a **domain expert** should validate explanations.

Churn Prediction: Conclusions

- Multiple options for **prediction + explanation**:
  - an LLM doing both,
  - interpretable model + explanation logic.
- Going further would not bring **new insights** for the course.
- Key takeaways:
  - choose the **right model complexity**,
  - always think about **how you justify** predictions.

A Note on Your Working Environment

- You have access to **GitHub Copilot** in the course environment (login required).
- As students, you can request free access via: `https://education.github.com/pack`

Preparing Data for LLM Fine-Tuning

- **input_ids** Token IDs for the entire sequence (system message + user instruction + expected assistant answer).
- **labels** Target tokens used for the loss. In instruction fine-tuning:
  - **Prompt tokens** = tokens from the **system** message and the **user** message → label = -100 (ignored in loss)
  - **Assistant tokens** = the model's answer → label = token ID (to be learned)

  This ensures the model learns only the **assistant output**.
- **attention_mask** Indicates which tokens the model should attend to:
  - 1 = real token
  - 0 = padding

Example Conversation (Chat Format)

Human–Readable Conversation

System

You are a helpful assistant.

User

Give me three tips to stay focused while studying.

Assistant

Here are three useful tips to improve concentration...

Python Chat Template Representation

```
messages = [
    {"role": "system",
     "content": "You are a helpful assistant."},
    {"role": "user",
     "content": "Give me three tips to stay focused while studying."},
    {"role": "assistant",
     "content": "Here are three useful tips to improve concentration..."}
]
```

After Applying the LLaMA 3 Chat Template

Rendered text (color-coded by role):

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>

You are a helpful assistant.<|eot_id|><|start_header_id|>user<|end_header_id|>

Give me three tips to stay focused while
studying.<|eot_id|><|start_header_id|>assistant<|end_header_id|>

Here are three useful tips to improve concentration...<|eot_id|>
```

Legend

- System segment (instructions for the model)
- User segment (the request)
- Assistant segment (what the model should learn to generate)
- Tags are colored with the role of the segment they belong to.

Mapping the Template to Labels (-100 vs Learned)

Color meaning for labels:

- Ignored (label = -100)
- Learned tokens (label = token_id)

Rendered text (color-coded by label):

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>

You are a helpful assistant.<|eot_id|><|start_header_id|>user<|end_header_id|>

Give me three tips to stay focused while
studying.<|eot_id|><|start_header_id|>assistant<|end_header_id|>

Here are three useful tips to improve concentration...<|eot_id|>
```

Multi-turn Conversation After LLaMA 3 Chat Template

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>

You are a helpful assistant for Python programming.<|eot_id|><|start_header_id|>user<|end_header_id|>

How can I create a list of numbers from 1 to 5 in
Python?<|eot_id|><|start_header_id|>assistant<|end_header_id|>

You can use list(range(1, 6)).<|eot_id|><|start_header_id|>user<|end_header_id|>

And how do I print each number?<|eot_id|><|start_header_id|>assistant<|end_header_id|>

You can loop over the list:  for x in numbers:  print(x).<|eot_id|>
```

Multi-turn: Which Tokens Become Labels?

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>

You are a helpful assistant for Python programming.<|eot_id|><|start_header_id|>user<|end_header_id|>

How can I create a list of numbers from 1 to 5 in
Python?<|eot_id|><|start_header_id|>assistant<|end_header_id|>

You can use list(range(1, 6)).<|eot_id|><|start_header_id|>user<|end_header_id|>

And how do I print each number?<|eot_id|><|start_header_id|>assistant<|end_header_id|>

You can loop over the list:  for x in numbers:  print(x).<|eot_id|>
```

Dataset & Instructions

Dataset: MG-ShopDial[1]
MG-ShopDial is a **multi-goal e-commerce dialogue** dataset with **64 human–human conversations** (2,196 utterances) where users naturally mix product search, recommendations, and item-related Q&A. Dialogs were collected through a coached protocol and annotated with **goals** and **intents**.

TP Instructions

- Load and explore the MG-ShopDial dataset in the notebook.
- **Fine-tune a pre-trained LLM** for conversational e-commerce.
- **Evaluate** the fine-tuned model using relevant metrics (perplexity, BLEU/ROUGE/BLEURT, qualitative checks).
- Compare with the **non fine-tuned** model.
- Test the model on **new, realistic examples**.
- Reflect on potential **improvements** for the chatbot.

[1] `https://raw.githubusercontent.com/iai-group/MG-ShopDial/main/MGShopDial/MGShopDial.json`