Large Language Models in Data Science

Week 6: Hackathon – Applied LLM Projects

Sebastian Mueller
Aix-Marseille Université
2025-2026

Session Overview

Morning (Hackathon briefing)

1. Recap of course concepts (Weeks 1–5)
2. Hackathon rules and grading
3. Project scope: what fits into one day
4. Team formation and logistics

Full-day Hackathon

- Choose a problem and dataset
- Design an end-to-end LLM / RAG pipeline
- Implement, test, and iterate
- Prepare a short live demo and summary

Why a Hackathon?

- Move from **toy examples** to an **end-to-end project** in a realistic setting.
- Combine techniques from the whole course: tokenization, embeddings, HF models, prompting, classification, RAG.
- Practice **teamwork** around LLMs and data: dividing roles, coordinating experiments, merging ideas.
- Experience trade-offs between ambition and scope under strict time constraints.

Evaluation & Final Grade

- The hackathon is the **final evaluation** of the course.
- Final score out of 20 is based on four criteria (5 points each), as in the README:
  - **Use Case & Applications (5 pts)**: relevance, originality, potential future usefulness.
  - **Code Cleanliness (5 pts)**: readability, organization, modularity, reproducibility.
  - **Lecture Integration (5 pts)**: use of course concepts (prompting, RAG, embeddings, classification, etc.).
  - **Presentation (5 pts)**: clarity, conciseness, visuals, explanation of design choices.
- Your hackathon score is your **official course grade**.

Teams & Logistics

- Work in groups of **1 to 5** students (recommended: 3–4).
- Each team produces:
  - a small but coherent **project repository or notebook(s)**,
  - a short **live demo** of the system (or a recorded run, if needed),
  - a **5–7 minute presentation** covering problem, approach, and lessons learned.
- Use any combination of tools seen in the course:
  - Hugging Face, Gemini API, scikit-learn, sentence-transformers, simple RAG stacks.
- You are encouraged to reuse and adapt code from the labs (with proper structuring and documentation).

What Makes a Good Project?

- **Clear problem statement**: for whom is the system built, and what decision does it help?
- **Grounded in data**: use real or realistic data relevant to AMU, Marseille, or data science practice.
- **Course concepts in action**:
  - thoughtful prompts, embeddings, RAG, or classification models,
  - not just a UI wrapped around an API.
- **End-to-end story**: ingestion $\rightarrow$ processing / retrieval $\rightarrow$ model $\rightarrow$ output.
- **Simple but solid**: small scope, but robust, well-structured, and easy to demo.

Scope for One Day

- Aim for a **minimal viable assistant**, not a full product.
- Prefer:
  - one or two well-implemented workflows,
  - over many incomplete features.
- Reuse:
  - lab pipelines (classification, RAG) as starting points,
  - prompts and templates tested earlier in the course.
- Keep infrastructure light:
  - local notebooks, small datasets, API calls,
  - no need for full web deployment if time is short.

Idea 1: AMU Data Science Programme Assistant

- **Goal**: help prospective or current students explore the MAS / Data Science track at AMU.
- **Data**:
  - official programme pages (like those used in the Week 5 lab),
  - additional documents: FAQs, course descriptions, internship info.
- **Pipeline**:
  - ingest and chunk web pages / PDFs,
  - build a BM25 + embedding index with metadata (campus, year, language),
  - use a RAG chain to answer questions about courses, prerequisites, careers.
- **Extensions**:
  - multilingual Q&A (French / English),
  - citations and links back to source sections,
  - a small evaluation set of typical student questions.

Idea 2: Data Science EDA Copilot

- **Goal**: assist a data scientist in exploring a tabular dataset (e.g., churn, housing, or a public Kaggle dataset).
- **Workflow**:
  - user provides a CSV and a question (e.g., “What drives churn in this dataset?”),
  - the system generates Python code snippets for plots / tables using prompting techniques from Week 3,
  - execute the code, collect results, and ask the LLM to summarise findings in natural language.
- **Course concepts**:
  - prompt templates for code generation and summarisation,
  - structured outputs (e.g., JSON with “steps”, “code“, “insights”),
  - maybe a simple classification model from Week 4 as a baseline.
- **Focus**:
  - reproducibility (scripts/notebooks can be re-run),
  - clear separation between data, code, and LLM outputs.

Idea 3: Student Support Inbox Triage

- **Goal**: help route incoming student emails to the right service and provide draft replies.
- **Data**:
  - a synthetic dataset of student enquiries (motivated by Week 4 intent classification),
  - labels such as `timetable`, `exam`, `administrative`, `internship`, `other`.
- **Pipeline**:
  - baseline classifier (embeddings + logistic regression or zero-shot classification),
  - optional RAG component with a small FAQ corpus,
  - LLM prompting to generate a draft response and a recommended action.
- **Evaluation**:
  - accuracy of routing to labels (using a held-out test set),
  - qualitative assessment of suggested replies (are they precise and polite?).

Checklist for Today

- **1. Define the problem**:
  - target user, key questions, success criteria.
- **2. Select data & models**:
  - datasets, external documents, HF models, LLM endpoints.
- **3. Design the pipeline**:
  - retrieval / preprocessing, model calls, post-processing.
- **4. Implement the minimal version**:
  - one or two key workflows working end-to-end.
- **5. Polish & present**:
  - clean up code, add README, prepare a concise demo.

Final Remarks

- Use this day to **experiment** and **connect ideas** from the course.
- Keep the scope realistic, but do not hesitate to be creative in your design.
- Remember the evaluation criteria: clarity of use case, clean code, integration of course content, and a clear presentation.
- Most importantly: have fun building with LLMs and data.