Erste Schritte zur LLM-Architektur & Planung deines Memory-Systems

**1. Modellarchitektur grob anschauen**

* Die meisten offenen LLMs (Large Language Models) basieren auf der Transformer-Architektur.
* Der Quellcode (meist in Python, oft PyTorch) ist auf GitHub oder Hugging Face öffentlich einsehbar.
  + Beispiel-Links:
    - Llama 3: https://github.com/meta-llama/llama3
    - Mistral: https://github.com/mistralai
    - Phi-3: https://github.com/microsoft/phi-3

**2. Was kannst du heute Abend konkret tun?**

**a) Architektur & Code anschauen:**

* Öffne auf Hugging Face z.B. Llama 3 oder Mistral.
* Dort findest du Model Card, Architektur-Details, Configs und manchmal die Forward-Funktion (das Herzstück des Modells).
* Lies dir für den Anfang nur die Modellübersicht und die Config-Datei durch – das reicht für ein erstes Bild.

**b) Planen & Skizzieren:**

* Skizziere auf Papier oder digital:
  + Welche "Module" brauche ich für mein Memory-System?
  + Wo greift mein "Memory-Hack" an? Vor dem Tokenizer? Nach der Modellantwort?
  + Was kann ich extern (in Python) lösen, was müsste "im Modell" passieren?

**3. Erste Ideen sammeln**

* Schreibe dir auf:
  + Was willst du speichern? (z.B. ganze Konversationen, Fakten, wichtige Abschnitte)
  + Wie groß ist dein Ziel-Kontext? (z.B. 100k, 1 Mio Tokens)
  + Willst du eigene Trainingsdaten einbauen? Oder "nur" den Kontext clever verketten?

**Tipp zum Abschluss:**

* Du musst heute Nacht nichts programmieren – es reicht, Notizen zu machen und zu träumen!
* Die besten Ideen entstehen oft genau in solchen Momenten.

**Nächste Schritte (für später):**

* Modell lokal laden und ausprobieren
* Architektur im Code nachvollziehen (besonders Forward-Funktion & Tokenizer)
* Memory-Strategie in Python antesten

Bleib neugierig – das Abenteuer beginnt jetzt!