Let’s sew some RAGs

•••

Bits&Brains talks series

Let’s check our suits before diving...

- Model vs API vs APP
- LLM/SLM, local vs cloud
- fine tuning vs model adjustment
- Context
- tokens
- vectors/embeddings

We need to talk boy!

- Repost messages to keep conversation
  - platform agents and tools: less control, less privacy, out of the box solution
  - own history: higher privacy, more control, more work, require more maintenance
- Calculate context size
  - possibility to ensure correct context size
  - cut context, summarise context, manipulate model params like windows size
- Switching model during work based on need
  - cost reduction
  - possibility to adjust model per specific action
- Different languages
  - use multilingual model: not all languages still supported, lower performance in some cases
  - use translation for content: additional complexity, costs
  - support only selected languages

Data, beam us up!

- Need to extend model capabilities
  - knowledge cut reduction - keeping up to date context
  - private and dedicated data that should not be used for training
- Prompts that wrap user question
  - build section of context to add better structure
  - improve prompt with prompt techniques and direct model to specific needs
- Use good format that is readable for the LLM
  - build structure like xml
  - use markdowns to underline sections and important parts
- Context size issue
  - context = question + data to process + prompt

I have it all here... somewhere...

- Adding more data = need for mechanism on data selection
  - minimize context to what is needed
  - identify needs with llm for example by embeddings, nlp methods, etc.
  - data must be searchable
- Vector DB, metadata and mix search
  - built in DB ai processing: limited control, limited models selection, auto processing
  - external vector build: more control, more options, must be ensured in all places
  - dedicated vector db vs general dbs: mix search, number of search methods, optimisation
  - content in search engine vs externally kept: complexity of data, size of the data, format of the data
- Decision to use specific embedding model and future change

JSON, PNG, PDF comes to the bar...

- Real life = mix form of data
  - use different storages
  - not everything must be vectorized
- Preprocessing is the key
- Big documents: cut big content to smaller pieces, use summaries
- Audio to text, full transcription or summary
- Images:
  - Keep to search: models that supports it, descriptions of images
  - Process: use descriptions, image processing model, attach to the message returned
- Data from DB added to the context in selected format (json, markdown)
- Query build on data provided by model:
  - strictly define parameter and format of returned structure by model
  - validate everything

Practice makes perfect

- Prompt management - versioning
  - keep your prompt as templates in separate files
  - versioning of prompt manually or with tools
- Prompt management - testing
  - test your prompt outside the app first
  - Use meta-prompts to improve your prompt
  - use tools like LangSmith, Promtfoo
- Prompt techniques
  - zero-shot, one-shot, few-shots, chain-of-thought, self-reflection, self-criticism, decomposition, panel of experts, meta-prompting, etc.
  - use structures and split different areas of the prompt

But it works on my computer!

- Life = complex data and its processing
- User and access control
  - like in any other app
  - remember to treat agents as services with service account!
- LLM input and output validation like SQL query, final content, jail breaking
- Build system knowledge with new requests:
  - web scraping, db updates, based on actions
  - save user feedback and add it to prompts, use user results for prompt testing
- Cost tracking
- Evaluation of selected content and re-search
- Secure communication with APIs, add privacy/security guardrails
- Use llm on results of llm for example to validate content, to build knowledgebase

Agent smith where are you?

- Not only chat as input/output: emails, cron calls, services calls like in agentic system or service logic
- Plan actions, list of actions, actions selection validation
- Evaluation of results, adjusting plans
- Action use LLM like build description for the event in calendar, summary of docs creation... but action run by the code
- Logs and action tracking
- Ready to use frameworks like CrewAI, LangChain with LangGraph