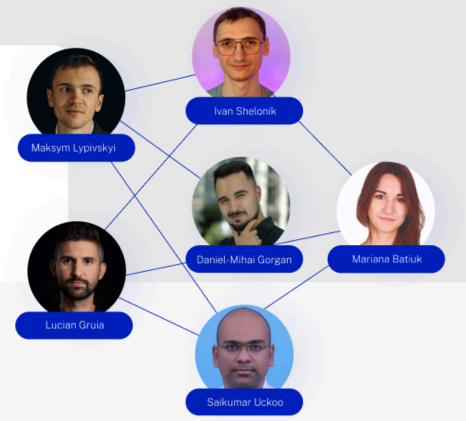
CIKLUM | SPEAKER'S CORNER

April, 16



Architecting Scalable AI RAG Systems:
From Startup to Enterprise.
A Live Coding Session

Structure:

- Video materials on the topic
- GitHub repositories
- Additional resources
- Q&A

Video library:

- recording “Architecting Scalable AI RAG Systems”
- recording “Neural network that can learn”
- recording “The Future of AI: Exploring OpenAI's products and services”
- recording “Building full stack AI chatbots with Java”

GitHub repositories:

- LLM Wrapper in Java
- JavaScript
- AWS (Python)
- Azure

Additional resources from Ciklumers related to the topic:

- Article Retrieval-Augmented Generation (RAG): A Leap in Generative AI by Lucian Gruia

Answers to the questions that were not addressed during the event:

Question	Answer
How expensive is it to run this on AWS?	Expensive - money We have used g5.xlarge 24GB GPU instance, which is $1.006 per/h https://aws.amazon.com/ec2/instance-types/g5/ Expensive - storage depends on the Model # of parameters and whether you want to use Quantization methods and which ones you want to use, etc.
I am working on an e-commerce website and will create a chatbot; can I use OpenAI or any other LLM model for production? Shall I buy any licenses?	- OpenAI costs you per request - no license is required to buy - HuggingFace open source LLMs - required to read license docs agreement regarding commercial license, but everything I have seen on HF is free to use even for commerce (Llama-2 is free entirely, you can use it for sure)
Can we do data analysis using RAG if I have my own data?	RAG can be used in data analysis where text retrieval and generation are required, primarily when deriving insights from textual datasets or automating reports. However, for more conventional data analysis involving structured data, statistical analysis, and visualization, traditional data analysis tools and frameworks are more appropriate. If you have your own data, consider your goals and whether you need text-based retrieval and generation or conventional statistical methods to determine the best approach.
How would we fetch the online data for training LLM models and scale the data size for training	Fetching online data for training Large Language Models (LLMs) involves using automated web scraping and data collection techniques to gather large volumes of text from various sources such as websites, forums, social media, and other publicly available platforms. Multiple strategies shall be employed to scale the data size: setting up distributed scraping systems to collect data concurrently from multiple sources, leveraging APIs where available to access structured data, and using data pipelines to process and store the incoming data efficiently. Data scaling also requires data preprocessing to clean, filter, and format the collected text, ensuring it is ready for training without

	redundancy or noise. Cloud-based storage and computing resources are utilized to manage large-scale data, allowing for distributed training and scalable data management across multiple nodes. Proper adherence to ethical considerations, including respect for data privacy and intellectual property rights, is crucial when fetching online data for LLM training.
I want to create a chatbot whose primary purpose is assisting an e-commerce website user with basic settings. I have a PDF that shows all the settings. Is the RAG chatbot able to answer the questions of 100 users simultaneously? Will it crash? Is a single purchase enough? Can we use LLM (OpenAI) for production and the public?	Yes, specifically --max-concurrent-requests=100 for TGI LLM inference, but ensure you have enough resources to handle it. Even though you have load weights, you still need GPU resources to generate. With limited input, it's hard to say exactly how many resources you will need. Try TGI inference from hugging face covered during the event; there are debugging and testing tools inside (read TGI docs), and you can try to generate such a load and figure it out exactly.
Can you explain the concept of quantization of LLM models for using models like Groq?	Quantization reduces the precision of the model's parameters from floating-point representations to lower-bit representations. Main quantization methods: - GPTQ (for GPU only) - GGML/GGUF - CPU + GPU - AWQ
Is there any roadmap to follow to learn about everything related to building apps with LLMs as a javascript/typescript developer?	RAG is good for structured data, but you can put your data in tuples or JSON. Explain the structure of tuples/JSON and in-context learning in the prompt to get good results.
What is better: llama7b or 13b quantized?	Depends on what fits you better: llama-7b - 7 billion parameters | less reasoning | less storage | faster llama-13b - 13 billion parameters | more reasoning | more storage | slower. But it's better to look into comparison plots in the articles, sites, etc.

How would you approach building a chatbot that does two things: 1) answers questions on existing data (so RAG pattern) but 2) can also do all the other things that LLMs do (for example summarize text)? How would you recognize the user's intent (if they want a RAG answer or a summary generated)?	Here, it would help if you had a mix of techniques to determine each prompt's context and the current conversation's state. In our presentation, on slide number 17, there is a diagram describing some components you need to use within your chatbot architecture to achieve this.
How do you enforce user access to underlying data in an RAG application? For example, a user may not have access to a file. However, over an RAG indexing pipeline, the document ended up in the vector database (e.g., salary information, data from another department, etc.).	You must implement isolation and user management at the app level, as different users have different access to specific knowledge bases. It is similar to traditional backend applications. However, one additional aspect is that you should leverage LLM capabilities when implementing knowledge base separation. Filtering data before creating the embeddings could enhance the system's security and ensure proper user-specific access.
How can we leverage LLM on CPUs without slowing down the process?	Model optimization techniques are vital in leveraging Large Language Models (LLMs) on CPUs without significantly slowing down the process. Compression methods like quantization and pruning can reduce the model's size and complexity, allowing for faster inference on CPU hardware. Quantization reduces the model's precision (e.g., from 32-bit to 8-bit), while pruning removes redundant parameters, making the model lighter and more efficient. Model distillation involves creating a smaller "student" model trained to replicate a larger "teacher" model's behavior and can also offer a compact version of LLMs that performs well on CPUs. Another approach to maintaining CPU efficiency is through optimized hardware utilization and efficient deployment strategies. Catch processing and pipeline architecture help increase throughput by processing multiple inputs simultaneously or offloading tasks to faster stages. Utilizing frameworks designed explicitly for CPU inference, such as ONNX Runtime or TensorFlow Lite, can significantly improve performance. Additionally, caching intermediate results and pre-processing data can reduce computational overhead, thus maintaining real-time processing capabilities. These combined strategies offer a way to work with LLMs on CPUs without sacrificing speed or functionality.

How can RAG innovate the Gaming Industry?	Retrieval-augmented generation (RAG) can innovate the gaming industry by enhancing narrative complexity, player interaction, and game content generation. By integrating RAG into game design, developers can create dynamic storylines that adapt to player choices, allowing for highly personalized gaming experiences. RAG can retrieve relevant information from large data sets or external sources, providing context-aware dialogues, character backstories, and interactive scenarios. This dynamic generation can lead to more immersive game worlds, with NPCs (non-player characters) responding contextually to player actions. Additionally, RAG can automate the creation of game assets, such as levels, missions, or quests, reducing development time and enabling more affluent, more varied gameplay. This ability to generate custom content and respond to player input in real time can transform how players engage with and experience games, leading to more engaging and interactive gaming experiences.
How can RAG influence the entertainment (iGaming) Industry?	Retrieval-augmented generation (RAG) can significantly influence the iGaming (internet gaming) industry by creating personalized and context-aware player experiences. By integrating RAG into online gaming platforms, developers can generate dynamic content tailored to individual users, such as personalized bonuses, game recommendations, and customized themes based on players' past activities and preferences. RAG can also enhance customer engagement and retention by providing real-time responses to player queries, thus improving customer support through chatbots or virtual assistants. This capability allows for a more interactive and responsive gaming environment, where the system can retrieve relevant information from large data sets to offer personalized suggestions, tips, or in-game events. Additionally, RAG can automate the generation of promotional content, emails, and marketing campaigns, enabling iGaming platforms to scale their outreach efforts with a higher degree of customization. By leveraging RAG's ability to process large amounts of information and generate contextually relevant outputs, the iGaming industry can create a more immersive and engaging experience, fostering stronger connections with players and encouraging prolonged gameplay.

How do you break into becoming an AI developer from a full-stack developer? Where to start from?	Becoming an AI developer, starting with full-stack development, is an excellent idea and usually works smoothly. You can begin by taking some courses on Data Engineering, Neural Networks fundamentals, and Generative AI. The next step would be to build some PoCs so you can have a way to practice. Later on, you will have to learn more about related topics continuously, and staying in touch with the community is the best way always to understand what is relevant.
How do we make use of re-ranking in RAG for better results?	Reranking within the RAG framework has substantial benefits. By implementing a reranking process, we can observe a notable increase in the relevance of retrieved information. A two-stage retrieval system offers the advantages of both scale and quality performance. Utilizing vector search enables efficient searching at scale. At the same time, reranking ensures that only the most relevant documents are prioritized, thus enhancing the overall quality of results within the RAG framework.
I have used cromadb as a vector db for my project for a rag system, but it saves the data in a file in my system. What are the options to keep the vectors in a centralized DB (I'm looking forward to either an in-memory storage like Redis or encrypted disk storage; i.e. if the data is stored in the disk, the DB should have a provision of keeping it secure)	To leverage the power of in-memory, you can always use the existing feature that Chroma SDK provides, but it wouldn't be persistent. Secondly, you can always spin up a chromadb service on a machine with disk encryption protection if you need encryption. https://learn.microsoft.com/en-us/azure/virtual-machines/disk-encryption-overview
Can RAG be implemented for images using any open-source image generation LLM?	Implementing Retrieval-Augmented Generation (RAG) for images with open-source image generation models is a developing area distinct from traditional text-based RAG. Open-source models like Stable Diffusion, DALL-E Mini, and BigGAN generate images from text or other inputs but don't inherently support RAG. To implement RAG for images, you must design a system that retrieves contextual information, such as metadata or text, to guide the image generation process. This could involve using retrieval mechanisms to fetch relevant data and then leveraging an image generation model to create visuals based on this context. Although open-source models do not directly support RAG, combining retrieval with image generation, a hybrid approach could achieve contextually informed image outputs.

How can AI benefit design in fighting climate change?	Artificial Intelligence (AI) can play a central role in designing solutions to combat climate change by optimizing energy use, improving resource management, and aiding in sustainable design practices. AI algorithms can analyze vast amounts of environmental data to identify patterns and trends, enabling better predicting climate-related events and improved planning for renewable energy deployment. In urban design, AI can optimize building energy efficiency, suggest eco-friendly materials, and reduce waste through intelligent recycling systems. Additionally, AI can aid in the design of sustainable transportation networks and optimize agricultural practices for reduced emissions. By integrating AI into climate change initiatives, designers can create smarter, more sustainable solutions that have a tangible impact on reducing carbon footprints and promoting a healthier planet.
How can I utilize AI in a b2b e-commerce solution?	In a scenario of a PLM engine, one can descriptively query their inventory to identify if an existing item matches a newly received requirement from a customer. Based on this, the closest matching SKU can be suggested. Eventually, it helps avoid the duplication of SKUs during the inventory.
What quantization techniques do you often rely upon?	Main quantization methods: - GPTQ (for GPU only) - GGML/GGUF - CPU + GPU - AWQ (the newest one).