2024 IEEE 7th International Conference on Multimedia Information Processing and Retrieval (MIPR) | 979-8-3503-5142-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/MIPR62202.2024.00108

Building a generative AI showroom for foundation models with different modalities

Benny Stein, Niklas Beck, Daniel Becker, Dennis Wegener
Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS
Sankt Augustin, Germany
{niklas.beck, daniel.becker, benny.joerg.stein, dennis.wegener}@iais.fraunhofer.de

***Abstract*—Multimodal foundation models represent a novel field of artificial intelligence that is gaining a lot attention and has a lot of potential in many application domains. In this work, we present how to build a generative AI showroom for single- or multimodal foundation models, which allows for the exploration and demonstration of the capabilities of these models. It provides a platform for users to interact with and understand the potential applications of generative AI technology across different modalities such as text, image, and audio. By showcasing the capabilities of these models, it helps in fostering innovation, collaboration, and understanding within the AI community. Additionally, the generative AI showroom serves as a valuable educational resource for both researchers and non-technical individuals, helping to bridge the gap between academia and industry. We provide details on the architecture and design decisions and present the showroom's features, performance and extensibility.**

***Index Terms*—LMMs, LLMs, foundation models, showroom, LLMOps, model serving**

I. Introduction

In the last years, generative AI and foundation models have gained a lot of attention, especially since ChatGPT was released. Foundation models were adopted first in the Natural Language Processing and Understanding (NLP/NLU) domain [1]. Nowadays, the interest has spread across a lot of other domains, hence showing the huge demand for foundation model-based AI solutions. Today, there are a lot of artifact types available. Closed-source foundation models and generative AI services are typically released via commercial APIs or in public cloud environments, whereas the open-source counterparts are made available via checkpoints on platforms such as, e.g., Hugging Face[1]. In both cases, a high-performance and cost-effective provisioning of GenAI services is difficult and production-ready on-premise solutions are not available (as of today). In this work, we outline how to structure and build a self-contained and on-premise platform for inference with multimodal (text, image, audio, embeddings and combinations thereof) foundational models to comply with aspects of data privacy, access management, trustworthiness, IT security, and – perhaps most importantly – usability for a broad range of downstream research as well as business scenarios. We give an overview on other existing platforms and demos, demonstrate the features and use cases of our foundation model showroom, and present our technical architecture and design decisions. Our instance is actively used by AI researchers and in workshops for companies just starting with adopting GenAI for their businesses.

[1]https://huggingface.co/

II. Related Platforms and Demos

In prior work, we already presented an NLU showroom for demonstrating NLU features like semantic similarity, entity linking, etc. [2]. Here, we now focus on related platforms and demonstrations of foundation models. Today, many platforms for accessing large language models and more general foundation models are available. Among them are, e.g., the following:

- OpenAI[2]: The most popular platform providing ChatGPT, several versions of GPTs, DALL·E and Sora for text, image and video generation as a service based on closed source models (non-public).
- Amazon Bedrock playgrounds[3]: A platform for testing inference on different models before using and deploying them in an application (non-public). On top, PartyRock[4] provides a code-free playground for building AI apps based on Bedrock.
- NVIDIA AI Playground[5]: This playground allows testing models from a catalogue via an individual demo UI.
- Databricks AI Playground[6]: Playground to test, prompt and compare different LLMs (non-public).
- Vercel AI model comparison[7]: This platform is centered around an SDK for comparing different models that also targets simple development of Java-/TypeScript interfaces.
- Hugging Face: one of the largest collections of open-source models, including an inference API and a UI for testing individual models.

Besides platforms where models are actually hosted, there are also some platforms which are just gateways to other providers or models, such as Kong AI Gateway and MLflow Deployments Server. These gateways aim at simplifying the comparison and replacement of LLMs by providing a more unified interface. Kong AI Gateway[8] currently supports the

[2]https://openai.com/
[3]https://docs.aws.amazon.com/bedrock/latest/userguide/playgrounds.html
[4]https://partyrock.aws/
[5]https://www.nvidia.com/en-us/research/ai-playground/
[6]https://docs.databricks.com/en/large-language-models/ai-playground.html
[7]https://sdk.vercel.ai/
[8]https://docs.konghq.com/gateway/latest/get-started/ai-gateway/

2770-4319/24/$31.00 ©2024 IEEE
DOI 10.1109/MIPR62202.2024.00108

providers OpenAI, Cohere, Azure, Anthropic, and some self-hosted models. A gateway that can be set up locally in minutes is the MLflow Deployments Server[9], where the providers[10] are specified by a simple configuration file.

It should be noted, however, that the list of platforms is incomplete and rapidly changing. The capabilities of these platforms are constantly changing, too, and due to the fast-paced community, the snapshot described here is likely to be outdated soon.

III. Showroom Features and Extensibility

Our showroom for foundation models (named "Fountain") provides a platform to demonstrate the versatility and capabilities of advanced AI systems in an integrated and interactive way, thereby enhancing understanding and trust in potential applications. It includes a variety of self-hosted models with different modalities, listed in Table I. In addition to these models, some models from OpenAI and a specific Azure OpenAI instance are also available.

We currently support the modalities t2t, t2e, t2i, t2a, a2t, where t, e, i and a stand for text, embedding, image and audio, respectively. t2t covers any LLM, t2e embeds text in a vector space and is thus important for retriever systems, and t2i allows for image generation models such as Stable Diffusion (open-source competitor of Midjourney and DALL·E). The modality t2a can be used for speech synthesis models, while a2t covers speech recognition models, transcription models and audio chatbots. We encountered a large palette of common use cases for models of the latter type, such as quickly formulating longer text passages such as letters or project reports without a keyboard or transcribing meetings. We have previously included the true multimodal models CLIP [8] and FLAVA [9] for vision-text alignment in our platform, but removed them in favor of other requested LLMs.

The modalities correspond roughly to the tabs in the user interface shown in Figure 1 - the only exception being t2e models, which we deemed not relevant for inclusion in the user interface, and only made them accessible via an API. This API is another key element of the platform and described in detail later in Section IV-B3. The main benefit of providing a dedicated API is its extensibility: It allows for larger workloads and in general much more traffic inside the system. Moreover, applications can be easily built on it. Many tools and frameworks are now available to incorporate Foundation models in software systems, or in general to operationalize them. Often, the first step is to wrap specific API calls to a model inside of derived class of a specific framework, such as Langchain or LlamaIndex for LLMs – just to name two of the most prominent and mature ones. After this rather simple step, the development of the remaining LLM-based application is almost identical as in the case that commercial LLM providers are used.

[9]https://www.mlflow.org/docs/latest/llms/deployments/index.html

[10]currently supported: OpenAI, MosaicML, Anthropic, Cohere, PaLM, Azure OpenAI, AI21 Labs, Amazon Bedrock, Mistral, and self-hosted models via HuggingFace TGI and models served via MLflow's model servers

IV. Technical Architecture and Design Decisions

A. Requirements

To deploy our platform, the basic requirements are a host machine with Docker and moderate CPU requirements, capable of hosting the databases, the Identity Access Management (IAM) system, API, and frontend (no GPUs are required for this machine). Typically, the most expensive requirement is a (in our setting separate) machine with a relatively recent GPU. Depending on the models and model backends to be used, we would recommend at least one GPU that is capable of hosting the required Foundation models without any quantization. In case of up to four 7B parameter models, a single NVIDIA A100 or H100 GPU is more than sufficient. With the use of quantization techniques, even less GPU capacities can suffice.

In principle, one can use different machines for different components, such as databases, webservers and backends. We decided to only separate the model backends and the IAM from everything else. This simplifies secure communication, as not all requests leave the host machine.

A Kubernetes-based deployment is possible, too, but probably less interesting in many scenarios as setting up a Kubernetes cluster is much more complicated in an on-premise setting than setting up Docker.

B. Overall Architecture

The overall architecture is shown in Figure 2 and described in detail in the following.

1) User interface: The user interface is built on top of Gradio[15] and mounted inside a FastAPI application server. It provides different tabs for Text Generation, Automatic Speech Recognition, Speech/Voice Synthesis and Image Generation, see Figure 1. In each of these tabs, the user can select from a list of models with suitable modality and interact with them. The frontend communicates with the IAM for authentication and with the API for the model access and inference.

2) Backend: The model backends are defined by nodes, which themselves consist of an identifier, a host address (with port), and a type. The type declares the interface that the model backend provides. Typical choices are `kserve`, `tgi` or `openai`, where the first one refers to the widely used kserve protocol[16], `tgi` refers to the API provided by HuggingFace's Text Generation Inference[17], and `openai` refers to any API that is compatible with the one provided by OpenAI (including Azure OpenAI). Depending on the specific needs and model backends used, more types can be added as required. Currently only `kserve` is used for models with modalities other than text and embedding.

A node corresponds to a serving backend, which can serve a single or multiple models. For example, if HuggingFace's TGI is used, then one node serves exactly one model. Other model serving backends, such as the NVIDIA Triton Inference

[15]https://www.gradio.app/

[16]https://github.com/kserve/kserve/blob/master/docs/predict-api/v2/required_api.md

[17]https://huggingface.github.io/text-generation-inference/

TABLE I
MODELS CURRENTLY ACCESSIBLE IN THE GENAI SHOWROOM

Model	Ref.	Input	Output
Meta's Llama 2 13b & 70b chat	[3]	text	text
MistralAI's Mistral-7B-Instruct-v0.2	[4]	text	text
MistralAI's Mixtral-8x7B-Instruct-v0.1	[11]	text	text
StabilityAI's Stable Diffusion SD-XL 1.0	[5]	text	image
OpenAI's Whisper-large-v3	[6]	audio	text
primeLine's Whisper-large-v3-german	[12]	audio	text
NVIDIA's FastPitch (en-US)	[7]	text	audio
Meta's MMS text-to-speech model (German)	[13]	text	audio
SentenceTransformers all-mpnet-base-v2	[14]	text	embedding

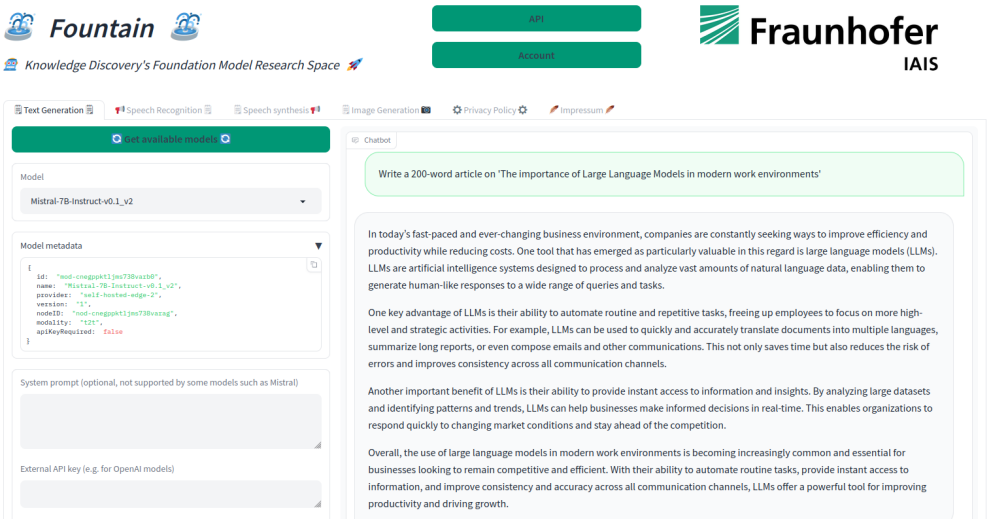

Fig. 1. GenAI Showroom User Interface

Server, are capable of serving multiple models with independent interfaces simultaneously. Note that the NVIDIA Triton Inference Server follows the `kserve` protocol, but provides many useful extensions to it. Among them are extensions for dynamically (un-)loading models, streaming, and binary in-/outputs. Additionally, the option to integrate TensorRT-LLM engines into the Triton Inference Server received attention for its outstanding inference performance[18].

External providers are a special kind of node and there is exactly one node for OpenAI and Azure OpenAI. We could incorporate other external providers, but typically the implementation of an adapter per provider is necessary.

3) API: The API server is implemented in Golang. It provides clients with a standardized interface to the various node protocols as described in Section IV-B2 by adapting the inference requests. It supports inference requests for all modalities listed in Section III. In addition, all inference requests can be executed synchronously or asynchronously. Asynchronous requests are placed in a queue (one queue per model) and processed successively in order to control resource consumption. The inference results are kept in a cache until they are retrieved by the user or the retention policy takes effect. The overhead caused by the API in comparison to the direct use of the nodes is in the sub-millisecond range. For a dynamic configuration of the accessible models, additional admin capabilities are CRUD options for both nodes and models, thus supporting faster access to the newest models.

4) Security and Access Management: We use SSL/TLS for the inter-network communication. If a node does not support https out-of-the-box a TLS termination proxy is used. Access Management for the user interface is realised via requesting party tokens (RPTs), whereas the API can be accessed with basic authentication. In the API, the requesting party's (an API user or the frontend on behalf of a user) roles determine whether a model or results can be accessed or not. We decided to define a role per model, but having a role per project might be more suitable in other scenarios.

5) Databases: The previously mentioned services are complemented by a set of databases. We suggest the following two database instances: Redis for caching results per user, global node and model configurations; a PostgreSQL database for storing user information. This second database is only accessed directly by the Identity Access Management system. An obvious extension of our platform would be to promote

[18] see https://github.com/premAI-io/benchmarks for a comparison with other inference engines

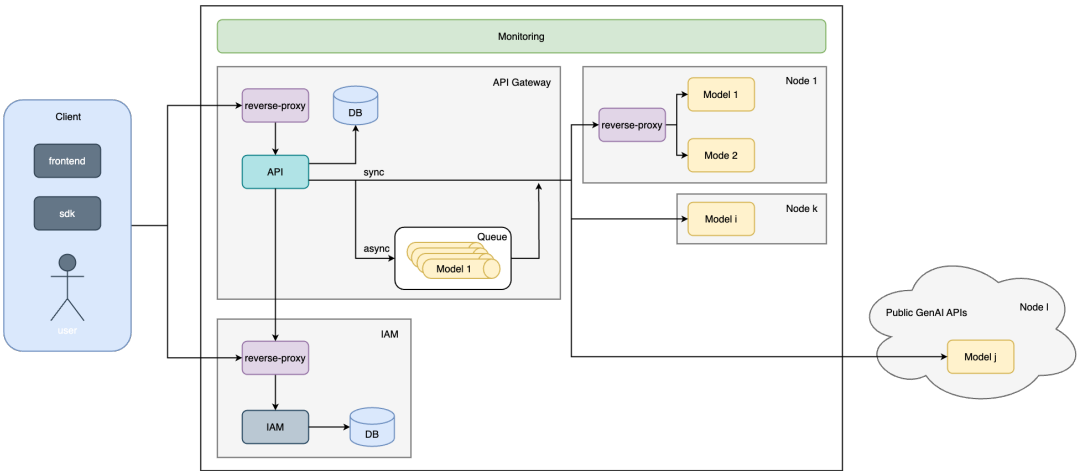

Fig. 2. GenAI Platform Architecture

foundation models to Retriever-Generator-systems. If doing so, the addition of a vector database is reasonable.

6) Monitoring: To properly monitor the whole platform, we provide health and metrics endpoints for the frontend and the API server. Many model backends expose health and metrics endpoints, too, e.g., TGI and NVIDIA Triton Inference Server.

The metrics endpoints are not exposed, but are consumed by a Prometheus instance. We then use Grafana dashboards to visualize all of the collected metrics to gain insights about traffic, cost, energy consumption, GPU usage, up-/downtimes of specific services, and other metrics such as usage of specific models for specific users, groups and projects.

V. Conclusion

In this work, we showed how to set up a showroom for foundation model inference capabilities. Our foundation model showroom is a cutting-edge demonstration that can run on-premise as well as containerized in cloud environments. With its extensive range of supported models and modalities, it offers a comprehensive solution for showcasing the latest advancements in AI. It allows for UI-based and API-based access and provides robust monitoring and access management capabilities to ensure security and control over access and usage. Looking ahead, the addition of trustworthy LLMOps demonstrations, RAG support, and advanced content-based monitoring will further enhance capabilities of this platform.

Acknowledgment

This work was funded by the Fraunhofer Cluster of Excellence »Cognitive Internet Technologies« CCIT and by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) through the project OpenGPT-X (project no. 68GX21007D).

References

[1] G. Paaß and S. Giesselbach, Foundation Models for Natural Language Processing: Pre-trained Language Models Integrating Media, ser. Artificial Intelligence: Foundations, Theory, and Algorithms. Springer International Publishing, 2023.

[2] D. Wegener, S. Giesselbach, N. Doll, and H. Horstmann, "Introducing the NLU Showroom: A NLU Demonstrator for the German Language," in 3rd Conference on Language, Data and Knowledge (LDK 2021), ser. Open Access Series in Informatics (OASIcs), D. Gromann, G. Sérasset, T. Declerck, J. P. McCrae, J. Gracia, J. Bosque-Gil, F. Bobillo, and B. Heinisch, Eds., vol. 93. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, pp. 28:1–28:9. [Online]. Available: https://drops.dagstuhl.de/entities/document/10.4230/OASIcs.LDK.2021.28

[3] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom, "Llama 2: Open foundation and fine-tuned chat models," 2023. [Online]. Available: https://arxiv.org/abs/2307.09288

[4] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mistral 7b," 2023. [Online]. Available: https://arxiv.org/abs/2310.06825

[5] D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Müller, J. Penna, and R. Rombach, "SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis," 2023. [Online]. Available: https://arxiv.org/abs/2307.01952

[6] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," 2022. [Online]. Available: https://arxiv.org/abs/2212.04356

[7] A. Łańcucki, "Fastpitch: Parallel text-to-speech with pitch prediction," 2021. [Online]. Available: https://arxiv.org/abs/2006.06873

[8] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning Transferable Visual Models From Natural Language Supervision," 2021. [Online]. Available: https://arxiv.org/abs/2103.00020

[9] A. Singh, R. Hu, V. Goswami, G. Couairon, W. Galuba, M. Rohrbach, and D. Kiela, "FLAVA: A Foundational Language And Vision Alignment Model," 2022. [Online]. Available: https://arxiv.org/abs/2112.04482