IBIMA Publishing
Communications of the IBIMA
https://ibimapublishing.com/articles/CIBIMA/2026/572159/
Vol. 2026 (2026), Article ID 572159, 19 pages, ISSN: 1943-7765
https://doi.org/10.5171/2026.572159

Research Article

Doku-Assist: Proactive Knowledge Retrieval for Service-Desk Agents: A Feasibility-Study on On-Premise LLMs for Data Privacy and Compliance

Jerome AGATER[1], Halil EGE[2], Ammar MEMARI[3] and Jorge MARX GÓMEZ[4]

[1]Applied Informatics, Jade Hochschule, Wilhelmshaven, Germany
[2]Carl von Ossietzky Universität Oldenburg, Germany
[3]Applied Informatics, Jade Hochschule, Wilhelmshaven, Germany
[4]VLBA, Carl von Ossietzky Universität Oldenburg, Germany

Correspondence should be adressed to: Jerome AGATER; jerome.agater@jade-hs.de

Received date:21 November 2025; Accepted date:19 January 2026; Published date: 11 March 2026

Abstract

In medium-sized organizations, frequent turnover of first-level support agents can lead to challenges for new agents, who struggle to discover existing and relevant documentation that would help solve user issues due to inexperience. Consequently, these agents escalate tickets to second-level support professionals, increasing their workload. A proactive knowledge discovery and assistance system targeting first-level service desk agents could help by analyzing tickets using a large language model (LLM) and then finding and presenting relevant documentation utilizing Retrieval-Augmented Generation (RAG) techniques. However, when working with cloud-based LLMs on inference tasks involving sensitive information, data sovereignty is compromised, and there is a risk of confidential content from tickets being leaked, as local information is transmitted to the cloud for processing. To address this issue, we constructed a system based on local LLMs so that the operation of the system does not compromise the privacy and confidentiality of ticket content and wiki documentation, keeping all sensitive data on-premise and secure. Our system, Doku-Assist, proactively finds and presents documentation to first-level support agents, thereby assisting with issue resolution without replacing the human agent. It integrates a DokuWiki-derived knowledge base with the ticket system Znuny. For the evaluation of our system, we used artificial tickets, documentation, and customer issues (to address privacy concerns) based on real-world experience. A second-level support agent was tasked with assessing the utility of the developed user interface as well as the documents proactively discovered and presented, concluding that the found documents presented by the Doku-Assist are useful to proactively fill the knowledge gap of new first-level service desk agents. We conclude that data privacy and law compliance can be achieved by utilizing local LLMs.

Keywords: LLM, Data Privacy, Local AI, Proactive System, RAG Integration

Introduction

When integrating AI systems based on Natural Language Processing (NLP) into organizations, chat interfaces quickly gained popularity following the media response to ChatGPT. The back-and-forth in chat interfaces allows untrained users to progressively and intuitively refine the responses of a large language model (LLM). Effective integration of such a chat interface into a workflow involves users formulating good queries, i.e., prompts, to the LLM, or it can require longer interactions with the chat interface to refine answers. Once the query's topic exceeds the domain of the models training data set, no further useful knowledge can be extracted, and the model might generate increasingly misleading information. Furthermore, if the LLM behind the chat interface or the entire application is hosted externally, sensitive data might leave the sovereignty of the organization and be at risk of information leakage, raising data privacy and other confidentiality concerns. Legal matters, such as the General Data Protection Regulation (GDPR), might disallow such usage altogether.

Similar to small- and medium-sized companies, the decentralized IT management of Faculty 2 at the University of Oldenburg handles the IT needs of more than 500 computer users, various research projects, work-groups, and productive server systems, taking care of deployment, servicing, procurement as well as license-, change-, incident- and problem-management. Issues are managed in a ticket system, with traditionally a two-level hierarchy of support agents utilizing the system. The first-level support had recruited students with limited experience and frequent staff turnover due to students finished studying, while the second-level support has been filled by experienced computer scientists, who have additional responsibilities besides user support. Due to the temporary nature of the students' employment, gained experience is lost, resulting in problems with knowledge transfer and finding existing knowledge.

An installation of the wiki software DokuWiki serves as the knowledge base in which the senior staffers document the various topics and solutions for problems not readily found by consulting a search engine. First-level support agents can access this source, however problems with knowledge discovery appeared, such as the students needing to know that knowledge for a specific topic is available at all and needing to know the correct terms to successfully search and find the existing information. Since the topics are so various, no good tagging system could be derived. Furthermore, students often lack the technical vocabulary to identify pages containing relevant information. Therefore, first-level agents often reverted to escalate a ticket to a second-level support agent, drawing resources away from topics with higher priority. The wiki content could be made accessible to an LLM, so it could query data on its own (either via a RAG system or using the model context protocol (MCP)). However, querying a chat interface by manually formulating a question and pasting information for every ticket takes time and does not necessarily lead to truthful results.

Rather than having first-level support agents look in the knowledge base for available information, then come back empty handed (or take a lot of time interacting with a chat interface), a proactive system could search for documents concurrently to the agents accessing the tickets, gathering relevant information, and pointing the agents to existing documents with relevant information, proactively. Due to the confidential nature and the protection of communication, however, no ticket content or knowledge base data may leave the premise to be processed outside. This requires a solution for on-premise processing of the information.

Data Sovereignty

Since the start of the LLM revolution in 2022 with the reveal of ChatGPT by OpenAI, the currently dominant form of interaction for users with pre-trained models is via chat interfaces. Typically, the applications powering these interfaces are either hosted web-applications running on cloud computing platforms, such as Azure Cloud or AWS, or their native counterparts running on mobile devices or workstations. In either case, the applications send the textual, voice/sound, or image content to application programming interface (API) endpoints served by backends running on the cloud servers in data centers of the LLM-system providers.

Control, Privacy, and Maintaining Sovereignty over Own Data

While the data being sent back and forth between an LLM and a client system are usually encrypted via secure transport protocols such as HTTPS or QUIC, both via TLS, the receiving end which hosts the LLM typically needs to decrypt the data for processing, therefore holding the data at least temporarily on the provider's hardware system in the clear. Once decrypted, they can potentially be siphoned by malevolent entities or accidentally be revealed/leaked[1]. The decrypted content can also be stored permanently in databases or serialized in storage volumes, allowing access and retrieval at a later date and time. Depending on the jurisdiction, legal means might be employed by interested third-parties or governmental institutions and authorities to force providers to allow access to such stored data troves.

Once in the hands of others, the data are at risk of leaks, unauthorized duplication, repackaging and further distribution, thereby escaping any control of the original owners of the data. This poses a concern for users aiming to control their own data and thriving for privacy of their sensitive information, but is also of high relevance for institutions aiming for compliance of their own data governance rules as well as privacy laws. We term the concept of maintaining sovereignty over own data *"data sovereignty"*. Depending on the location of the users and the data, different legal frameworks might apply. ChatGPT's originator OpenAI and other LLM inference providers reside in the USA. Moving data from one jurisdiction to another imposes additional challenges that need to be addressed when the data are transmitted to data-centers within the USA for inference. Regulations regarding privacy and personal information such as the General Data Protection Regulation (GDPR) in the EU but also the California Consumer Privacy Act (CCPA) and California Privacy Rights Act (CPRA) in the USA affect the handling of sensitive personal data. Furthermore, LLM-service providers might train their models on the submitted data, subsequently unintentionally revealing sensitive information by releasing the trained model later on, from which parts of the training data can potentially be extracted as Carlini et al. demonstrated in the case of the GPT-2 series in 2021 (Carlini et al., 2021), with the corresponding implications for its privacy (Huang et al., 2024); see also the work of Karamolegkou et al. (2023).

Context

From on-premise to cloud

Since at least the early 2010s, there has been a steady trend to move from existing local IT setups towards cloud setups, even though critical challenges to privacy exist (Akremi and Rouached, 2021; Ghorbel et al., 2017). Both IT and software departments of companies may advocate for this move due to reasons like perceived cost reduction and easier management[2]. Cloud systems typically provide an easy way of integrating and provisioning new resources, i.e., instead of the procurement, installation, and setup of actual hardware in an on-premise data center. In cloud environments like Amazon Web Service (AWS) or Microsoft Azure, further resources can be booked via software/web interfaces on demand. In contrast to this, ChatGPT[3], the first large language model to become available and be used mainstream, debuted as a cloud-based web-application. Consequently, the system has always been a cloud system, operating within the operating hardware context of OpenAI. While earlier and less capable versions of transformer-based language models were free to use, more complex LLMs like GPT 3.5 were simply not available to be run locally. Companies wanting to adopt capable LLMs into their business processes had to use the externally hosted models or rely on weaker models. Consequently, organizations considering integrating LLM-based AI systems into their workflows and tool repertoire had to accept a necessary reliance on cloud-based/third-party-run LLMs – or not be able to profit from the potential of LLMs.

More-Private Cloud Computing: Attempts at private computing without full encryption

Companies, such as Apple Inc., who formulate the privacy of their users' data as a goal, have started to create combined hardware/software solutions to enhance privacy protection by not relying on remote cloud services of third parties. In the case of Apple, specific hardware for the execution of LLMs is integrated with efficient smaller LLMs. The operating system[4] integrates the AI functionality into services providing the functionality to applications running on it. From Foundational LLMs, reduced LLMs can be derived, which require less computational resources, but are typically also less capable. Even though they may lack broad generality, they can be useful for niches. For the creation of the smaller models, a larger model generates text, which is used as the knowledge and source material to train the smaller LLM. The data sets generated do not contain the same wealth and depth of the training data of the larger models. Since one of the goals of the smaller models is to be smaller, the model internal weight space available for knowledge storage is necessarily a lot smaller as well. To alleviate the shortcoming, Apple has created Private Cloud Compute (PCC) (Gunter et al., 2024), a stateless system that supplants the on-device models with external inference from a larger foundation model. In this schema, the data are also encrypted in flight, but the encryption is end-to-end towards a specific target compute node in the PCC infrastructure. Therefore, the data stay encrypted until they reach the final processor node responsible for the actual inference computation. Apple claims to not be able to access said data, and that the data are not even accessible to data center personnel with administrative access. This claim rests on the operating system for the compute node being notarized by Apple. However, nothing technical hinders Apple from authorizing a version of the operating system with backdoor access.

As another approach in the context of mental health, Sarwar proposes to fine-tune LLMs to preserve data privacy and comply with Health Insurance Portability and Accountability Act (HIPAA) and GDPR laws via a federated learning framework (Sarwar, 2025), crucially noting the risk of data leakage during the fine-tuning process. In this approach, training datasets available on different clients are used to create fine-tuning weights on each client individually; the generated weights then get sent to a central server, which performs the fine-tuning of base/foundational LLMs via Low-Rank Adaptation (LoRA), resulting in a fine-tuned model aggregation; this newly derived model can then be sent back to the clients for use. Sarwar notes that federated learning approaches do not perform as well as direct approaches (Sarwar, 2025).

Fully Private Cloud Computing: Homomorphic Encryption Schemes for Privacy-Preserving Inference and Training

To protect sensitive information in transit and at rest, cryptography is commonly used. A compelling array of open-source and proprietary solutions exists – from HTTPS to encrypted hard disks, using symmetric and asymmetric encryption. However, once the data need to be accessed for computation inside a program on an execution node, the encrypted data are usually decrypted to be processed. In a homomorphic encryption scheme, the encrypted data remain encrypted while being processed by homomorphic operations, i.e., operations that work on encrypted data directly providing a result that stays encrypted without ever decrypting anything. A fully homomorphic system (FHS) supports encrypted execution of arbitrary functions by implementing and providing homomorphic operations, which as a set allow for Turing-complete programs. In 2009, Craig Gentry proved such a system to be possible using ideal lattices (Gentry, 2009). While this scheme itself is impractical, the groundbreaking work allowed other FHSs to be developed. As the data are never decrypted, such a system can be used as a crucial component for implementing a Zero Trust (ZT) computation environment. Some cloud providers offer fully homomorphic encryption for some of their cloud offerings. For example, in 2021, Google open-sourced their transpiler and later created a tool-chain by developing an intermediate representation for homomorphic encryption called HEIR[5] based on MLIR (Lattner et al., 2021), to create a MLIR dialect suitable for all (modern) FHE schemes. One promising goal of the project is to integrate with TensorFlow[6]. However, the computationally expensive matrix-matrix multiplications as well as the extensive model size move simple transpilation further to the horizon. In 2021, Chillotti et al. showed that efficient homomorphic inference is possible in general for Deep Neural Networks by other means (Chillotti et al., 2021). Some companies have started to commercialize FHSs[7] to provide different services requiring ZT, such as confidential voting systems. Active research is in progress to directly allow homomorphic encryption with LLMs, including private fine-tuning. Efforts from research teams such as Rho et al. to introduce encryption-friendly LLM architectures (Rho et al., 2025) already show efficient and secure fine-tuned processing of natural language data is possible for some real-world language models, e.g., in their case, an encrypted BERT-style transformer was implemented using LoRa. Given the current trend of lowering computation time and reducing communication cost (see, for example, the work of Pang et al. (2023), optimizing the overall execution), we can expect this path to privacy-preserving HE inference to improve further. At the moment, setting up and using an HE-based solution is out of scope for most interested users.

Towards On-Premise LLM Services

A new option became available in 2023 when a team from Meta Inc. released the open-weight[8] LLaMA model in February 2023[9], which provides performance on par with the performance of ChatGPT3 (Touvron et al., 2023), and shortly later, when Mistral AI announced[10] an open-source LLM in September 2023, finally allowing to run capable models locally. Since then, families of LLMs have been released and updated by a variety of other companies, e.g., Alibaba (Qwen), Anthropic (Claude), DeepSeek (DeepSeek R1), Google (Gemma), IBM (Granite), Microsoft (Phi), and xAI (Grok). Finally, OpenAI itself released the open-weight models gpt-oss-120b and gpt-oss-20b for local inference with an Apache 2.0 license in August 2025[11].

Even though a pool of model families is available now, constrained resources can be a restricting factor in running LLMs locally. LLMs typically require powerful hardware with regards to the Graphical Processing Units (GPUs) or other AI accelerator cards, i.e., data-center GPUs like NVIDIA H100/H200, providing the matrix multiplication and other computing features required for efficient AI execution. To run a state-of-the-art LLM locally, specific hardware might have to be obtained, as the existing hardware of an organization or researchers is often not powerful enough to allow the execution of demanding models. However, commodity hardware, e.g., GPUs for gaming PCs, already allow the execution of well-performing LLMs. Furthermore, of-the-shelf consumer systems employing unified memory, i.e., a system where CPU and GPU can access the same memory without the need for copying and transferring data between memory regions, allow access to large amounts of VRAM (e.g., 512 GB) and thereby the efficient execution of LLMs with a high number of parameters. This can reduce the barrier to entry for organizations seeking to leverage LLM-based AI systems. Once a local setup is established, the information is never endangered to be siphoned away or unacceptably accessed by keeping data under local lock and never moving them from on-premise storage to cloud or other third-party computing infrastructure. This strategic shift to on-premise LLMs allows organizations with sensitive data to embrace and build upon the LLM AI trend. We define such a strategy as a local-only strategy, i.e., a strategy where no cloud-based inference providers are used and all inference work is done on-premise on machines under the users' exclusive control and possession, leading to data sovereignty for the users.

Architecture of a typical Local-Only System

When decomposing a typical local-only system, we can identify interconnected and interdependent base components, namely the concrete LLM being employed; the inference engine/model runner, which executes the neural architecture encoded in the LLM on CPUs or GPUs; possibly a networking layer providing the LLM inference as a service (and also possibly aggregating load balancing, authorization, and authentication, etc.); and finally the interfacing or host application, which makes use of the LLM inference capabilities to derive the actual value for a use-case. Furthermore, when employing a RAG-architecture to reduce hallucinations (Lewis et al., 2020; Zhang and Zhang, 2025), we can find an embedding model to vectorize data in a knowledge base and a data store for the storage of generated vector embeddings, such as a vector database. If the system utilizes the model context protocol (MCP), such a system integrates with further services mediated with an MCP framework handling the protocol interactions (Hou et al., 2025). While external providers host their services typically in a cloud environment with a complex network mediated and load-balanced API wrapping the inference service encapsulating the LLMs executed on the hardware/cloud of the provider – e.g., OpenAI with their various GPT offerings, self-hosting an LLM on-premise opens up the possibility to run the inference engine(s) on the same local system, possibly embedded into the interfacing application itself. We can identify the following taxonomy for main communication architectures of data-sovereign systems:

embedded architecture:
the LLM engine and models are integrated within the same application, e.g., as a library;

local client-server architecture:
the interfacing application communicates with the LLM engine through inter-process communication or a network mediated API;

distributed client-server architecture:
the interfacing application communicates via network with LLM inference engines running LLMs on.

Different open-weight and open-source components are available to select for the assembly of such a local system.

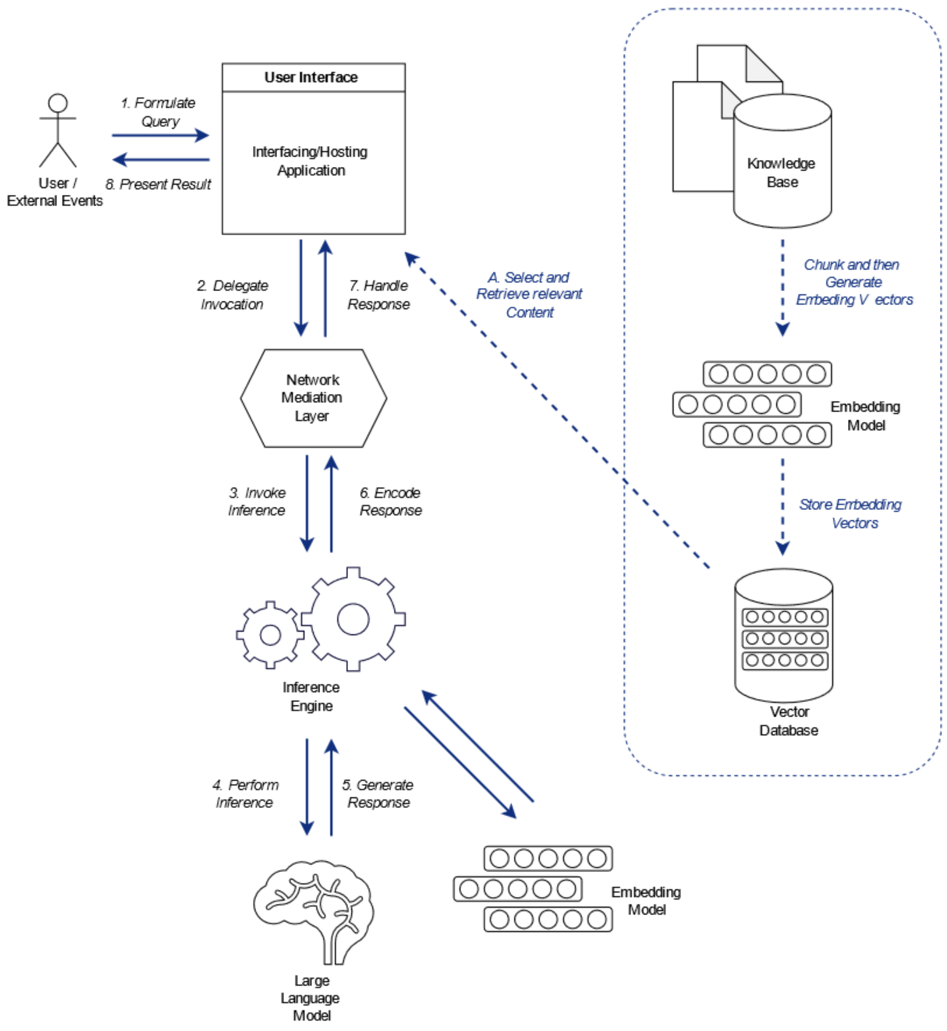


Fig 1: Query-Response Flow in a Local-Only LLM System. A flow is initiated by users formulating a query or system events (1). The Interfacing/Host Application combines the query with internal state, system prompts, and other information into a context then forwards the prompt to a network middleware layer (2). The layer invokes the inference engine, which hosts the LLMs and in case of RAG systems possibly also embedding models (3). The inference engine uses the LLM to perform the actual inference based on the prompt (4). The LLM generates a response for the inference engine (5). The inference engine encodes the responses for the network layer (6). The network layers return the result to the hosting application to handle the response of its initial query (7).

Interfacing/Hosting Application

An interfacing or hosting application is the component which interacts with the user or a surrounding system. The application takes user or system input and directs interactions or workflows through an UI or system-interfacing APIs, controlling the interaction with and the invocation of the LLM, mediating its tool-use (if any), handling the processing and defining the general program flow.

Large Language Model

Different LLMs can be downloaded in the form of encoded model weights and neural architectures. Foundational models are the basis from which smaller, more application-oriented models can be derived. Typically leveraging unsupervised learning on unlabeled big corpora of textual data[12], foundational models for Natural Language Processing (NLP) are being trained at scale, i.e., thousands of nodes and terabytes of data. Such foundational models have learned the general structure, syntax, and semantics of natural languages, providing the flexibility to be used for many general NLP tasks. To improve their suitability for specific tasks, such models can be fine-tuned, i.e., retrained on domain-specific content. Enabling the model to follow user-given instructions can be achieved by training the model on instruction-response pairs, e.g., using Reinforcement Learning with Human Feedback (RLHF) (Christiano et al., 2017). One example of this workflow is the foundational model General Purpose Transformer (GPT) from OpenAI and its fine-tuned ChatGPT and InstructGPT versions. While OpenAI hasn't released its newer models (3 and on) for download, other capable and open-source/open-weight[13] foundational and instructed models can be retrieved from model hosting sites, with HuggingFace establishing itself as "the GitHub for models"[14], a repository and community platform for collaboration on models.

Arguably (Fierro et al., 2024), large language models contain encoded knowledge from their training data (see for example as a key-value memory described by Geva et al. (2021) for transformers), however, due to the probabilistic nature of response generation, generated output may exhibit unwanted or surprising properties, an important aspect that should not be forgotten while constructing LLM-based systems. We differentiate between several classes of problems with the generated output from LLMs, which often get subsumed under the general term hallucinations, see Huang et al. (2025) for an alternative taxonomy and extensive survey. LLMs are prone to hallucinations, i.e., generations that look plausible to a human reader but refer to subjects that do not exist (Ji et al., 2023; Huang et al., 2025). When tasked to support their work with references, LLMs may and do generate references and citations of non-existing work (Zuccon et al., 2023), e.g., non-existing articles in scientific journals, laws and law paragraphs, norms, books, or persons, for which numerous studies have been done, e.g., by Buchanan et al. (2024). The output may or may not be correct, but hallucinated references obviously cannot contribute to or support the veracity of the output. In the context of programming, LLMs may refer to and use names and imports of functions, methods, and libraries to solve a problem, but the used building blocks have no real-world counterpart, so the code fails when run, e.g., Dou et al. (2024) investigated the problems with LLM generated code in detail. Similar to hallucinations are wrong generations, where the LLM presents facts as being correct and continues to elaborate on its train of thought leading to further wrong generations or hallucinations (Zhang et al., 2024), even though the stated information is in fact not congruent with any factual or real-world data. Interestingly, the internal state of the LLM might indicate awareness of incorrect text being generated (Azaria & Mitchell, 2023). Furthermore, nonsensical output, where the generation of an LLM does not reflect the task implied from the input at all, may present as gibberish/unreadable sequences of characters or the generation stops without actually having generated an output related to the implied task with the output appearing meaningless. Finally, there are dangerous generations, where the LLM may suggest actions to be taken that are harmful, where the LLM acts aggressive or otherwise unacceptable towards the user (Gehman et al., 2020), e.g., one of Gemeni's recent generations was covered in the news (Clark and Mahtani, 2024), or generates output containing hazards/forbidden information, like construction plans for bombs or toxic or abuse-related substances. Due to the statistical nature, dangerous generations could potentially also appear inside of regular output, e.g., requested construction help for a turbine may also contain an unwanted detonation device, or assistance with a chemical formula when executed in a laboratory may produce a toxic agent as a by-product. As almost all models[15] inherently may produce such unwanted output due to training data, it is imperative to take this into account and apply guardrails, see for example Padhi et al. (2024).

The race to increase performance and output quality is improving the capabilities quickly, so consulting performance leaderboards can help select the right model for the designated application to create. Different leaderboards are evaluating LLMs for specific tasks, e.g., coding, and rank them according to their performance. However, Alzahrani et al. (2024) caution against blindly relying on the ranking, as some of the rankings have been shown to be not robust against perturbations.

Model Runners as Inference Engines

To perform inference with an LLM locally, the neural networks represented by its architecture and weights need to be loaded into the GPU or CPU, and the queries and responses need to be managed. Furthermore, to interconnect and utilize the LLM, implementing an API around it to infer output from queries and return the output to the rest of our application in a structured manner is considered best practice. Different projects and organizations have started to work on creating solutions for such local execution of the models. Conceptually, such a runner can either be included in the application, i.e., a library running in the same process, or the runner can be an external program, acting as a mediator by providing an access mechanism for the exchange between the main application and the LLM. For external execution, modeling an HTTP API according to OpenAI's API (OpenAI, 2024) allows tools and code written with the OpenAI API in mind to work with other local LLMs without the need for modification.

To experiment with models, prompts, and their generated output, user-friendly LLM environments have proven to be very useful, e.g., LM Studio[16] or Silly Tavern[17]. With LM Studio, an OpenAI compatible API can be spawned for programmatic access to models (Studio, 2024), easing experimentation for production use. However, production applications are typically deployed to a dedicated server or installed as applications and should not rely on an additional desktop application as a critical component. Common choices for running an inference service locally and thereby keeping data sovereignty range from custom created runners (e.g., a Python script using PyTorch or HuggingFace's Transformers library) to optimized runner services integrating performance optimization techniques (e.g., vllm or onnx).

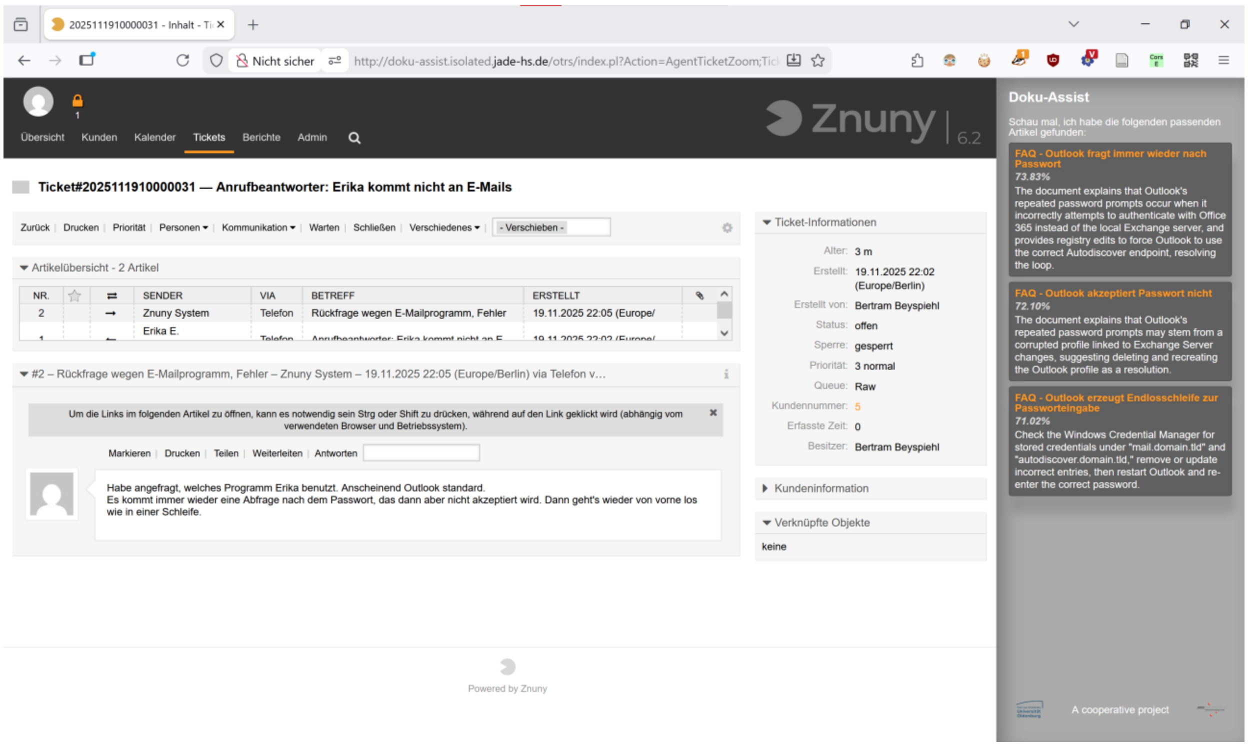
Custom: A dedicated local LLM runner can be constructed manually by leveraging Python and the well-known PyTorch, Flux, or TensorFlow libraries to construct the layer and network architecture of a transformer-based LLM and then loading the weights. However, this requires a lot of work and in-depth knowledge of the libraries and LLM's architecture.

Hugging Face Transformers: With the Python library[18] Hugging Face transformers (Wolf et al., 2020), this process is abstracted away into a unified interface. Furthermore, transformers provides methods for downloading, configuring, quantization, and fine-tuning models. While transformers is generally imported as a library, it's also possible to create an OpenAI-compatible API by writing the endpoints with a web framework oneself, e.g., by leveraging the web framework FastAPI. Furthermore, the transformers library is the basis for notable workflow frameworks like LangChain19.

LLaMA Derived Runners: The initial LLaMA release from Meta contained code to load models and run inference via the PyTorch-based Python code[20]. The Python implementation was quickly rewritten in C++ to form the llama.cpp project[21], providing a speedup against Python and allowing CPU-based inference. llama.cpp provides bindings for many languages including Python, Rust, Go, Ruby, Java, and JavaScript. Using llama.cpp as the foundation, Ollama[22] adds a user-friendly interface for the management functionality to download, update, and organize model files, as well as the startup and running of inference. Another project leveraging llama.cpp is LLamafile[23], which combines it with Cosmopolitan Libc to create a universal executable from an LLM, so it can be started as a regular application on multiple platforms without modification.

vLLM: Optimized for bigger workloads, vLLM[24] provides a complete LLM inference and serving engine performing efficiently at large-scale deployments by implementing request batching as well as a virtual memory inspired paging mechanism for a subset of the memory involved in LLM execution called PagedAttention. The technological principles of which are explained in a research paper (Kwon et al., 2023). Furthermore, vLLM includes other optimizations to improve GPU memory usage and support distribution in multiple GPU setups, as well as techniques to improve speed and efficiency like Speculative decoding and Flash attention. The project has a very active community, hosting events and meetups.

Feasibility Study: The Proactive System Doku-Assist



Ticket#2025111910000031 — Anrufbeantworter: Erika kommt nicht an E-Mails

Artikelübersicht - 2 Artikel

NR.	☆	⇄	SENDER	VIA	BETREFF	ERSTELLT
2		→	Znuny System	Telefon	Rückfrage wegen E-Mailprogramm, Fehler	19.11.2025 22:05 (Europe/
1		←	Erika E.	Telefon	Anrufbeantworter: Erika kommt nicht an E	19.11.2025 22:02 (Europe/

#2 – Rückfrage wegen E-Mailprogramm, Fehler – Znuny System – 19.11.2025 22:05 (Europe/Berlin) via Telefon v...

Um die Links im folgenden Artikel zu öffnen, kann es notwendig sein Strg oder Shift zu drücken, während auf den Link geklickt wird (abhängig vom verwendeten Browser und Betriebssystem).

Habe angefragt, welches Programm Erika benutzt. Anscheinend Outlook standard.
Es kommt immer wieder eine Abfrage nach dem Passwort, das dann aber nicht akzeptiert wird. Dann geht's wieder von vorne los wie in einer Schleife.

Ticket-Informationen

Alter:	3 m
Erstellt:	19.11.2025 22:02 (Europe/Berlin)
Erstellt von:	Bertram Beyspiehl
Status:	offen
Sperre:	gesperrt
Priorität:	3 normal
Queue:	Raw
Kundennummer:	5
Erfasste Zeit:	0
Besitzer:	Bertram Beyspiehl

Kundeninformation

Verknüpfte Objekte

keine

Doku-Assist

Schau mal, ich habe die folgenden passenden Artikel gefunden:

FAQ - Outlook fragt immer wieder nach Passwort
73.83%
The document explains that Outlook's repeated password prompts occur when it incorrectly attempts to authenticate with Office 365 instead of the local Exchange server, and provides registry edits to force Outlook to use the correct Autodiscover endpoint, resolving the loop.

FAQ - Outlook akzeptiert Passwort nicht
72.10%
The document explains that Outlook's repeated password prompts may stem from a corrupted profile linked to Exchange Server changes, suggesting deleting and recreating the Outlook profile as a resolution.

FAQ - Outlook erzeugt Endlosschleife zur Passworteingabe
71.02%
Check the Windows Credential Manager for stored credentials under "mail.domain.tld" and "autodiscover.domain.tld," remove or update incorrect entries, then restart Outlook and re-enter the correct password.

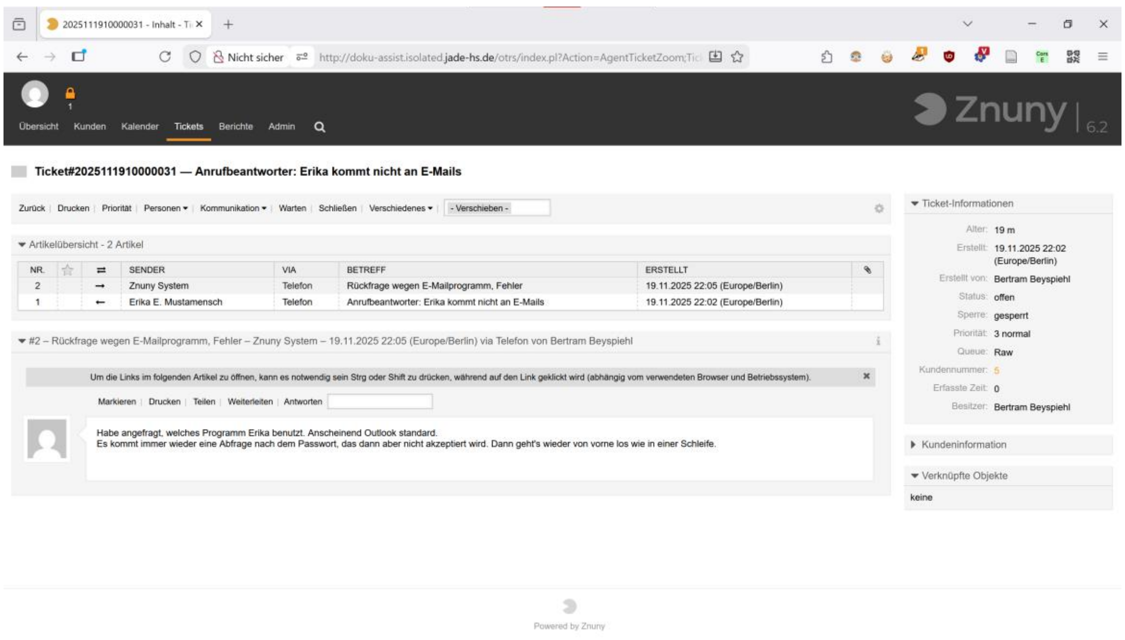
Fig 2: A screenshot of an active Doku-Assist session (right side) inside Znuny. When helpful documents are found in the knowledge base, the sidebar of Doku-Assist opens automatically, offering explanations why the results might be helpful to the human first-level support agent (descriptions switched to English for this screenshot).

A proactive system retrieves and presents relevant information before a user actively asks for it by anticipating the need for the information. LLMs can also be used for proactive conversational dialogue systems (Deng et al., 2025). Recently, LLMs have been successfully employed to offer proactive assistance in various scenarios, from coding (Zhao et al., 2025), in interactions while exploring data in natural language interfaces (Tabalba et al., 2025), to providing knowledge support in audio-centric media environments (Zhifei et al., 2025). We developed a proactive knowledge assistance system, integrated into the Znuny-based ticket solution, to alleviate the knowledge discovery problem for first-level support service agents and thereby reduce the load on second-level agents. In accordance with the goal of data sovereignty, our system runs on-premises, utilizing open-source LLMs and software. The system analyzes conversations in the tickets and then provides documents helpful for finding solutions to the issues described in these tickets, as well as the reasons why the provided documents might aid with the task at hand. The intent is for customers to have the (correct) feeling that they are communicating with a human. Therefore, the interaction is not driven by an LLM but rather by a human in the loop who can utilize the potentially helpful documents to formulate an answer. Our system is available as open-source.

Involved Components

While an OTRS system was used in the past, we decided to implement its successor, the ticket system Znuny. The knowledge base is maintained in an installation of DokuWiki. The embeddings from the RAG system of the plug-in aichat are used for knowledge retrieval in the proactive assistance system.

Ticket System Znuny



2025111910000031 - Inhalt - Ti

Nicht sicher http://doku-assist.isolated.jade-hs.de/otrs/index.pl?Action=AgentTicketZoom;Ti

Übersicht | Kunden | Kalender | Tickets | Berichte | Admin

Znuny 6.2

Ticket#2025111910000031 — Anrufbeantworter: Erika kommt nicht an E-Mails

Zurück | Drucken | Priorität | Personen | Kommunikation | Warten | Schließen | Verschiedenes | - Verschieben -

Artikelübersicht - 2 Artikel

NR.	☆	⇄	SENDER	VIA	BETREFF	ERSTELLT	
2		→	Znuny System	Telefon	Rückfrage wegen E-Mailprogramm, Fehler	19.11.2025 22:05 (Europe/Berlin)	
1		←	Erika E. Mustamensch	Telefon	Anrufbeantworter: Erika kommt nicht an E-Mails	19.11.2025 22:02 (Europe/Berlin)	

#2 – Rückfrage wegen E-Mailprogramm, Fehler – Znuny System – 19.11.2025 22:05 (Europe/Berlin) via Telefon von Bertram Beyspiehl

Um die Links im folgenden Artikel zu öffnen, kann es notwendig sein Strg oder Shift zu drücken, während auf den Link geklickt wird (abhängig vom verwendeten Browser und Betriebssystem).

Markieren | Drucken | Teilen | Weiterleiten | Antworten

Habe angefragt, welches Programm Erika benutzt. Anscheinend Outlook standard.
Es kommt immer wieder eine Abfrage nach dem Passwort, das dann aber nicht akzeptiert wird. Dann geht's wieder von vorne los wie in einer Schleife.

Ticket-Informationen

Alter:	19 m
Erstellt:	19.11.2025 22:02 (Europe/Berlin)
Erstellt von:	Bertram Beyspiehl
Status:	offen
Sperre:	gesperrt
Priorität:	3 normal
Queue:	Raw
Kundennummer:	5
Erfasste Zeit:	0
Besitzer:	Bertram Beyspiehl

Kundeninformation

Verknüpfte Objekte

keine

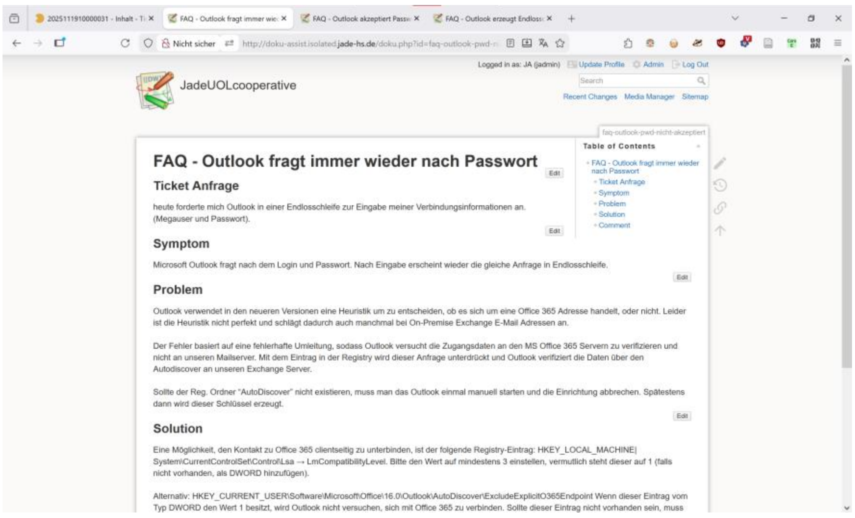
Powered by Znuny

Figure 3: The view of a first-level support agent inspecting a ticket. Typically, only one "article" (conversation message) in a ticket is visible at a time.

Znuny is a web-based ticket system for help- and service desks, allowing agents of an organization to work on and respond to help requests from customers (reporters) via telephone or e-mail via a generally user-friendly browser-based interface, as depicted in Figure 3. In 2023, Prastama et al. identified room for improvement in a case study with OTRS (Prastama et al., 2023) regarding the user interface, however. Originally developed and released as the Open Ticket Request System in 2001 by OTRS AG for Unix-like operating systems, new versions of the ((OTRS)) Community Edition were available as free software until early 2021 when OTRS AG announced the end of their open-source strategy.

The open-source fork Znuny25 was created by the OTTER Alliance and Znuny GmbH, supporting teams relying on the open-source OTRS variant to continue using an open-source version with support from a commercial vendor. Znuny is written in Perl. Like the OTRS variants, Znuny tracks tickets in the form of queues, helping to split incoming tasks into relevant areas of expertise. Znuny exposes a REST-based API via its configurable web-service interface, which allows flexible integration into and with external systems. Znuny allows the installation of plug-ins or packages, which extend the functionality of a Znuny installation. Integrating customizations is also possible by configuring additional CSS and JavaScript resources to be loaded depending on the context, e.g., such that a customization is only loaded into the viewing screen page of an agent.

Knowledge Base DokuWiki


202511191000031 - Inhalt - T
FAQ - Outlook fragt immer wie
FAQ - Outlook akzeptiert Passw
FAQ - Outlook erzeugt Endloss
Nicht sicher
http://doku-assist.isolated.jade-hs.de/doku.php?id=faq-outlook-pwd-n
Logged in as: JA (jadmin)
Update Profile
Admin
Log Out
JadeUOLcooperative
Search
Recent Changes
Media Manager
Sitemap
faq-outlook-pwd-nicht-akzeptiert
Edit

FAQ - Outlook fragt immer wieder nach Passwort

Ticket Anfrage

heute forderte mich Outlook in einer Endlosschleife zur Eingabe meiner Verbindungsinformationen an. (Megauser und Passwort).

Symptom

Microsoft Outlook fragt nach dem Login und Passwort. Nach Eingabe erscheint wieder die gleiche Anfrage in Endlosschleife.

Problem

Outlook verwendet in den neueren Versionen eine Heuristik um zu entscheiden, ob es sich um eine Office 365 Adresse handelt, oder nicht. Leider ist die Heuristik nicht perfekt und schlägt dadurch auch manchmal bei On-Premise Exchange E-Mail Adressen an.

Der Fehler basiert auf eine fehlerhafte Umleitung, sodass Outlook versucht die Zugangsdaten an den MS Office 365 Servern zu verifizieren und nicht an unseren Mailserver. Mit dem Eintrag in der Registry wird dieser Anfrage unterdrückt und Outlook verifiziert die Daten über den Autodiscover an unseren Exchange Server.

Sollte der Reg. Ordner "AutoDiscover" nicht existieren, muss man das Outlook einmal manuell starten und die Einrichtung abbrechen. Spätestens dann wird dieser Schlüssel erzeugt.

Solution

Eine Möglichkeit, den Kontakt zu Office 365 clientseitig zu unterbinden, ist der folgende Registry-Eintrag: HKEY_LOCAL_MACHINE| System\CurrentControlSet\Control\Lsa → LmCompatibilityLevel. Bitte den Wert auf mindestens 3 einstellen, vermutlich steht dieser auf 1 (falls nicht vorhanden, als DWORD hinzufügen).

Alternativ: HKEY_CURRENT_USER\Software\Microsoft\Office\16.0\Outlook\AutoDiscover\ExcludeExplicitO365Endpoint Wenn dieser Eintrag vom Typ DWORD den Wert 1 besitzt, wird Outlook nicht versuchen, sich mit Office 365 zu verbinden. Sollte dieser Eintrag nicht vorhanden sein, muss

Table of Contents
- FAQ - Outlook fragt immer wieder nach Passwort
  - Ticket Anfrage
  - Symptom
  - Problem
  - Solution
  - Comment

Fig 4: DokuWiki displaying an example wiki page of documentation

DokuWiki is a wiki-based content management software designed specifically for creating documentation (see Figure 4). Released in 2004 and continuously developed since then, DokuWiki is widely used by enterprises as well as organizations[26], including IT departments of universities[27]. DokuWiki can also be used to implement document management systems for ISO/IEC 27001:2022, with templates readily available from commercial sources. The Australian company BuiltWith[28] surveys websites on the Internet to identify the technology used to implement them. According to BuiltWith[29], DokuWiki accounts for 30.32% of wiki installations out of 61,930 detections, ranking second behind MediaWiki (which powers Wikipedia). DokuWiki is open-source software, with a commercial offshoot derived from it targeting enterprise users (Voigt et al., 2011). DokuWiki is written in PHP. As a data storage layer for its textual content, plain text files[30] are utilized instead of relational database management software, making the software more robust against infrastructure failure. This file-based approach simplifies importing and exporting of contained data, as well as allowing straightforward filesystem-based backup solutions. DokuWiki can be extended by a plug-in system, which allows customization and extension of its behavior and features.

Embeddings and RAG

DokuWiki can integrate a RAG-based AI chat over its own wiki data using the extension dokuwiki-plugin-aichat[31]. The GUI of the plug-in is implemented in JavaScript as a custom HTML element. The UI sends questions to a server-side backend API, awaits the response, and then appends the query and answer in its UI chat log. The API backend handles the queries by utilizing different inference providers over the network, e.g., calls to OpenAI or Ollama, making the system flexible regarding the specific LLM and inference engine, following the flow from Figure 1. The RAG system utilizes rendered wiki text (without the syntax) or, if a renderer is unavailable, the raw wiki-syntax-embedded text to generate chunks of 1,000 tokens. It then generates embedding vectors from the chunks using the backend. The chunks, their embeddings, and a reference to the originating page are then stored either in the vector databases Chroma, Pinecone, or Qdrant, or in the relational, file-based database SQLite. The system finds relevant chunks for user queries, then deriving its answers from the retrieved data like in a typical RAG system (Lewis et al., 2020).

Construction of Proactive Assistance System

Core Setup

For our implementation of the interface, as well as the actual application logic and workflow (i.e., the hosting application), we utilized TypeScript[32] as the implementation language due to its typing and tool support. Our application consists of multiple source files, and, since TypeScript does not run directly in browsers, we used the bundler Parcel[33] to transpile the entire web application, spread over multiple source files and directories, from TypeScript to JavaScript. Parcel also minimizes, optimizes, and bundles the application into just a few files for distribution. This enables us to inject and load our assistance application by configuring only two additional files to be loaded by the Znuny web interface, namely the compiled JavaScript and CSS, as shown in Table 1. We chose this integration path for its simplicity.

Table 1: Configuration settings of Znuny to integrate the two transpiled, optimized, and compressed files forming our extension.

Configuration Key	Added Value
Loader::Agent::CommonJS###000-Framework	thirdparty/doku-assist-0.1.1/index.js
Loader::Agent::CommonCSS###000-Framework	thirdparty/doku-assist-0.1.1/index.css

To retrieve the contents of the tickets, we used the web-service capabilities of Znuny. Znuny provides access to its controller classes, which implement its behavior via a flexible, configurable JSON-based web API. We used the recommended web-service configuration from OTRS. However, we extended the API configuration using the administrative interface of Znuny to accept HTTP POST calls for exchanging larger messages and HTTP OPTIONS calls necessary due to the CORS requirements from browsers. The official OpenAPI Spec/Swagger specification provided by OTRS[34] for the web-service API was used as the basis for generating a TypeScript API client implementation using the NSwag tool-chain[35].

On-Premise, Data-Sovereign Inference and Embedding

To host our LLMs locally, we utilized a spare system (see Table 2) running Ubuntu Linux as the hardware basis for our development. We chose Ollama (version 0.6.8) as the inference engine and web API interface for our local inference (see Section 4.2) due to its ease of use, wide adoption, and LLM management capabilities. Installation and setup were convenient and straightforward. The ai-chat plugin allows the configuration of OpenAI-API- and Ollama API-compatible local models. Integrating our on-premises Ollama API only required adding the system's URL and access configuration. Using the Ollama command-line interface, new models can be pulled from a remote repository and loaded for serving with Ollama's built-in API.

Table 2: Specification of the Hardware used to run the inference engine and its web-API

Case	Dell Tower
Processor	AMD Ryzen Threadripper PRO 5965WX 24-Cores
RAM	128 GB RAM
Storage	1 TB NVMe SSD
GPU	NVIDIA RTX 6000 Ada Generation, 48 GB

Open-Source Embedding and LLM Models

We downloaded and experimented with all the embedding models supported by the ai-chat plug-in, but settled on mxbai-embed-large, an open-source embedding model offering performance comparable to commercial models like text-embedding-3-large from OpenAI (Lee et al., 2024). Since mxbai-embed-large had already been tested successfully with and by DokuWiki, we did not evaluate unsupported, newer embedding models. We expect such models to offer better performance characteristics according to leaderboards and benchmarks such as the Massive Text Embedding Benchmark (MTEB); however, we have not yet had the chance to verify this assumption. As readily configurable options, the ai-chat plug-in supports common local chat model families such as LLaMA 3, Gemma 2, and Qwen 2.5. However, due to its instruction-following capabilities, support for CoT, permissive Apache-2.0 license, prior positive experiences with the model, and its generally good performance on our hardware, we decided to use Alibaba Cloud's Qwen3-32B. By patching the list of supported models, Qwen3 could be integrated as a configuration option as well.

Implementation of AI API

We initially experimented with the API of the ai-chat plug-in; however, the concrete structure of the question-response structure was not well aligned with our use case, as the system's response already processed the content of all potentially relevant documents into an answer, whereas we wanted a document-specific description of the reasons why a particular document was relevant for the first-level support agent. We patched the API, adding two new endpoints: specifically, an LLM proxy that allows us to query the LLM hosted in Ollama mediated through DokuWiki, and a document selection and retrieval endpoint based on the established RAG system (see Figure 5 for the JSON returned).

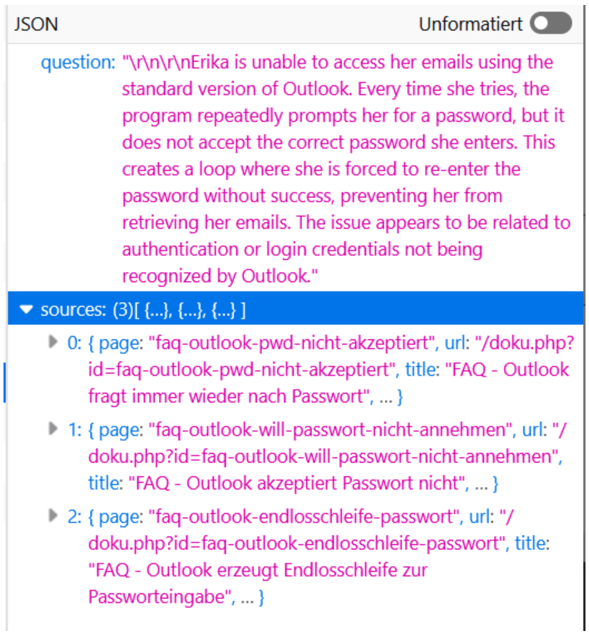
Implementation of LLM Workflow and Interfacing Application

Once the application is loaded into the view of an agent by Znuny, a check is performed to determine if the current page represents a ticket view. If that is the case, the system initiates the proactive workflow illustrated in Algorithm 1.

Algorithm 1 Pseudocode for Proactive Workflow

```
1:  procedure FINDHELPFULDOCUMENTS          ▷ Invoked on Agent Views for Tickets
2:      $ticketId \leftarrow$ EXTRACTCURRENTTICKETID
3:      $conversation \leftarrow$ RETRIEVECONVERSATIONPARTS(ticketId)
4:      $prompt_{summary} \leftarrow$ fill $PromptTemplate_{summary}$ with $conversation$
5:      $summary \leftarrow \mathrm{LLM}(prompt_{summary})$
6:      $candidateDocuments \leftarrow \mathrm{RAG}(summary)$
7:      $results \leftarrow array[\,]$
8:      for all $document \in candidateDocuments$ do
9:          $prompt_{assessment} \leftarrow$ fill $Template_{assessment}$ with $content(document)$
10:         $assessment_{document} \leftarrow \mathrm{LLM}(prompt_{assessment})$
11:         push $(document \cup assessment_{document})$ onto $results$
12:     end for
13:     $relevantDocuments \leftarrow \{doc \in results \mid$ assessment deems $doc$ relevant$\}$
14:     PRESENT($relevantDocuments$)
15: end procedure
```

For the current page, all the contents of the conversation parts of a ticket are retrieved using the web service configured in Znuny (1–3). The conversation parts (called "articles") are integrated into a prompt (4). The LLM hosted in our local inference engine is then queried using the ai-chat plugin proxy to summarize the conversation such that relevant words are retained, and missing terminology for technical terms is inserted (5). The summarizing response is used as a query vector in the RAG system provided by the ai-chat plug-in, using our extension API to find matching documents. The API returns metainformation about the documents as well as their contents (6), see Figure 5. For each of the matching documents (8), a prompt is constructed, containing the meta-information, the document, and the reformulated query with instructions to assess the usefulness and describe why the content is useful (9). This prompt is sent to the LLM for deciding whether the document can help the agent by supplementing instructions relevant to the conversation (10). The LLM responses are gathered and combined (11). If a document is deemed relevant, it is selected for presentation in the user interface (13). Finally, if any documents have been deemed relevant, the user interface opens and presents the relevant documents as well as the description (14), resulting in the sidebar opening as depicted in Figure 2. A link leads the agents to the DokuWiki page, where they can interact using the existing chat interface for further inquiries about the documents.



```
question: "\r\n\r\nErika is unable to access her emails using the standard version of Outlook. Every time she tries, the program repeatedly prompts her for a password, but it does not accept the correct password she enters. This creates a loop where she is forced to re-enter the password without success, preventing her from retrieving her emails. The issue appears to be related to authentication or login credentials not being recognized by Outlook."
sources: (3)[ {…}, {…}, {…} ]
  0: { page: "faq-outlook-pwd-nicht-akzeptiert", url: "/doku.php?id=faq-outlook-pwd-nicht-akzeptiert", title: "FAQ - Outlook fragt immer wieder nach Passwort", … }
  1: { page: "faq-outlook-will-passwort-nicht-annehmen", url: "/doku.php?id=faq-outlook-will-passwort-nicht-annehmen", title: "FAQ - Outlook akzeptiert Passwort nicht", … }
  2: { page: "faq-outlook-endlosschleife-passwort", url: "/doku.php?id=faq-outlook-endlosschleife-passwort", title: "FAQ - Outlook erzeugt Endlosschleife zur Passworteingabe", … }
```

Figure 5: The descriptive summary of the conversation in the ticket submitted for retrieval of matching documents and meta-information about found documents.

To implement the UI of our interfacing application, we chose React[36], a commonly used UI framework for JavaScript from Facebook (Ferreira et al., 2022). We did not need a state management framework (like MobX or Redux) since our user interface is mostly presentational without complex state changes visible to the user.

Initial Evaluation

For our evaluation of the system, we used the curated data corpus in the existing DokuWiki installation, augmented with information from the first-level support data of the IT Department (containing general information such as how to join the *eduroam* network, excluding any sensitive technical details). Since we cannot use real tickets containing privacy sensitive user data without users' consent, we manually generated anonymized tickets with conversations representative of real conversations. Finally, a second-level support agent was tasked with assessing the documents presented by the system as an acceptance test.

In our initial experiments, useful documents were generally presented, which would help with solving the tasks encoded in the conversations. In some instances, the AI incorrectly rejected documents that were deemed useful by the agent and in some instances the RAG system did not return documents due to chosen threshold settings or not matching vocabulary used. While offering good summarization and interpretation skills, the response time of Qwen3-32B often exceeded 1 minute, which is too long. However, as the process runs parallel to the agents reading and understanding the tickets, the perceived wait time was reported as acceptable.

Outlook

After acquiring the necessary permits to run an operational AI system within the user-facing support system, we aim to gather feedback from first-level support agents, e.g., by asking the agents to score the relevancy of the provided documents and their helpfulness in resolving the tasks outlined by the tickets. Once enough training data have been collected, we could automatically evaluate more embedding models to improve document retrieval using RAG optimization techniques. By applying methodology from prompt engineering, the summaries of the conversations in the tickets could offer an opportunity for further retrieval optimization. Finally, utilizing more powerful hardware can reduce the waiting time until the first documents are offered, making the system more convenient to use for professional users.

Conclusion

Artificial Intelligence, in the form of LLMs, can be used in more ways than just typical chatbots. A proactive system can add value for users without the need to engage directly with a chat interface, working instead effortlessly besides and on behalf of the users, utilizing the state-of-the-art inference and instruction following from open-source LLMs. Running such a system based on open-source components locally with open-weight/-source models allows practitioners to benefit from the LLM trend without sacrificing privacy or having to refrain from utilizing such a system due to privacy, secrecy, or trade secret considerations.

Acknowledgment

The work of the author Jérôme Agater was funded by the JadeProf project.

Notes

[1] e.g., on March 24, 2023, OpenAI revealed that, due to a flaw in a third-party library, messages from conversations of users were potentially visible to unrelated users, leaking the information content

[2] be it Hardware as a Service (HaaS), Platform as a Service (PaaS), or Software as a Service (SaaS), e.g., cloud-hosted databases

[3] ChatGPT announced in November 2022, see https://openai.com/index/chatgpt/, [Online], [Retrieved April 1, 2024], based on GPT-3.5

[4] i.e., macOS starting with version 15.1 Sequoia

[5] see HEIR: Homomorphic Encryption Intermediate Representation, homepage: https://github.com/google/heir

[6] See "Project Goals" on https://heir.dev/#project-goals

[7] e.g., Zama. https://www.zama.ai/, [Online], [Retrieved December 11, 2025]

[8] The term open-weight in the context of LLMs means that the parameters (or values of the matrices) making up a model's layers are freely and publicly available for use and modification by third-parties such as researchers or enterprises, similar to the term open-source which means that source code itself is (also) available.

[9] LLaMA 3, released in April 2024, see https://ai.meta.com/blog/meta-llama-3/, [Online], [Retrieved June 27, 2024]

[10] via a blog post on their website at https://mistral.ai/news/announcing-mistral-7b/, [Online], [Retrieved March 31, 2025]

[11] announced on their website at https://openai.com/index/introducing-gpt-oss/, [Online], [Retrieved August 8, 2025]

[12] e.g., crawls of the reachable internet, but also books, articles, and so on

[13] while the weights, the final model, may be open and available for download, the training data and the training regimen are often not, therefore describing such models as open-source might not be considered a correct term.

[14] https://huggingface.co, [Online], [Retrieved April 2, 2025]

[15] compare claims by IBM Research (Mishra et al., 2024)

[16] LM Studio on GitHub, https://github.com/lmstudio-ai, [Online], [Retrieved April 21, 2025]

[17] SillyTavern on GitHub, https://github.com/SillyTavern/SillyTavern, [Online], [Retrieved April 21, 2025]

[18] transformers on GitHub, https://github.com/huggingface/transformers, [Online], [Retrieved April 21, 2025]

[19] LangChain on GitHub, https://github.com/langchain-ai/langchain, [Online], [Retrieved April 23, 2025]

[20] original release on branch llama_v1, see LLaMA on GitHub, https://github.com/meta-llama/llama/tree/llama_v1, [Online], [Retrieved April 21, 2025]

[21] llama.cpp on GitHub, https://github.com/ggml-org/llama.cpp, [Online], [Retrieved April 21, 2025], based on the tensor library ggml, ggml on GitHub, https://github.com/ggml-org/ggml, [Online], [Retrieved April 21, 2025]

[22] Ollama on GitHub, https://github.com/ollama/ollama, [Online], [Retrieved April 21, 2025]

[23] Mozilla-Ocho LLamafile on GitHub, https://github.com/Mozilla-Ocho/llamafile, [Online], [Retrieved May 21, 2015]

[24] virtual large language model, originally developed at the Sky Computing Lab at UC Berkeley, vllm on GitHub, https://github.com/vllm-project/vllm, [Online], [Retrieved April 21, 2025]

[25] Znuny on GitHub, https://github.com/znuny/Znuny, [Online], [Retrieved November 10, 2025], https://www.znuny.org/de, [Online], [Retrieved November 10, 2025]

[26] e.g., the open router software OpenWRT https://openwrt.org/, [Online], [Retrieved October 17, 2025], and institutions like DAPNET https://hampager.de/dokuwiki/, [Online], [Retrieved October 17, 2025]

[27] Notable examples include the Berliner Hochschule der Technik, TU Clausthal, TU Darmstadt, and Jade Hochschule

[28] https://trends.builtwith.com/cms/wiki/traffic/Entire-Internet, [Online], [Retrieved October 17, 2025]

[29] Data last updated on October 17, 2025, checked October 20, 2025

[30] Consisting of UTF-8 encoded text structured with DokuWiki-flavored wiki syntax

[31] dokuwiki-plugin-aichat on GitHub, https://github.com/cosmocode/dokuwiki-plugin-aichat, [Online], [Retrieved October 1, 2025], developed by the original DokuWiki author Andreas Gohr

[32] TypeScript on GitHub, https://github.com/microsoft/TypeScript, [Online], [Retrieved April 30, 2025]

[33] Parcel-Bundler on GitHub, https://github.com/parcel-bundler/parcel, [Online], [Retrieved October 10, 2025]

[34] https://academy.otrs.com/de/doc/admin/processes-automation/web-services/web-service-documentation/, [Online], [Retrieved September 30, 2025]

[35] NSwag on GitHub, https://github.com/RicoSuter/NSwag, [Online], [Retrieved September 30, 2025]

[36] React on GitHub, https://github.com/facebook/react, [Online], [Retrieved October 14, 2025]

References

- Akremi, A. and Rouached, M. (2021), 'A comprehensive and holistic knowledge model for cloud privacy protection', J. Supercomput. 77(8), 79567988. [Online], [Retrieved Jun 2025], https://doi.org/10.1007/s11227-020-03594-3
- Alzahrani, N., Alyahya, H., Alnumay, Y., AlRashed, S., Alsubaie, S., Almushayqih, Y., Mirza, F., Alotaibi, N., Al-Twairesh, N., Alowisheq, A., Bari, M. S. and Khan, H. (2024), 'When benchmarks are targets: Revealing the sensitivity of large language model leaderboards', in L.-W. Ku, A. Martins and V. Srikumar, eds, 'Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)', Association for Computational Linguistics, Bangkok, Thailand, 13787–13805.
- Azaria, A. and Mitchell, T. (2023), 'The internal state of an LLM knows when it's lying', in H. Bouamor, J. Pino and K. Bali, eds, 'Findings of the Association for Computational Linguistics: EMNLP 2023', Association for Computational Linguistics, Singapore, 967–976.
- Buchanan, J., Hill, S. and Shapoval, O. (2024), 'Chatgpt hallucinates non-existent citations: Evidence from economics', The American Economist 69(1), 80–87.
- Carlini, N., Tramèr, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T., Song, D., Erlingsson, Ú., Oprea, A. and Raffel, C. (2021), 'Extracting training data from large language models', in '30th USENIX Security Symposium (USENIX Security 21)', USENIX Association, 2633–2650. [Online], [Retrieved Jun 2025], https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting
- Chillotti, I., Joye, M. and Paillier, P. (2021), 'Programmable bootstrapping enables efficient homomorphic inference of deep neural networks', Cryptology ePrint Archive, Paper 2021/091.
- Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S. and Amodei, D. (2017), 'Deep reinforcement learning from human preferences', in I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, eds, 'Advances in Neural Information Processing Systems', Vol. 30, Curran Associates, Inc.
- Clark, A. L. and Mahtani, M. (2024), 'Google AI chatbot responds with a threatening message: "Human Please die." — cbsnews.com', https://www.cbsnews.com/news/googleai-chatbot-threatening-message-human-please-die/. [Online], [Retrieved Apr 2025].
- Deng, Y., Liao, L., Lei, W., Yang, G. H., Lam, W. and Chua, T.-S. (2025), 'Proactive conversational ai: A comprehensive survey of advancements and opportunities', ACM Trans. Inf. Syst. 43(3). [Online], [Retrieved Jun 2025], https://doi.org/10.1145/3715097
- Dou, S., Jia, H., Wu, S., Zheng, H., Zhou, W., Wu, M., Chai, M., Fan, J., Huang, C., Tao, Y., Liu, Y., Zhou, E., Zhang, M., Zhou, Y., Wu, Y., Zheng, R., Wen, M., Weng, R., Wang, J., Cai, X., Gui, T., Qiu, X., Zhang, Q. and Huang, X. (2024), 'What's wrong with your code generated by large language models? an extensive study'. [Online], [Retrieved Jun 2025], https://arxiv.org/abs/2407.06153

- Ferreira, F., Borges, H. S. and Valente, M. T. (2022), 'On the (un-)adoption of javascript front-end frameworks', Software: Practice and Experience 52(4), 947–966. [Online], [Retrieved Jun 2025], https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.3044
- Fierro, C., Dhar, R., Stamatiou, F., Garneau, N. and Søgaard, A. (2024), 'Defining knowledge: Bridging epistemology and large language models', in Y. Al-Onaizan, M. Bansal and Y.-N. Chen, eds, 'Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing', Association for Computational Linguistics, Miami, Florida, USA, 16096–16111.
- Gehman, S., Gururangan, S., Sap, M., Choi, Y. and Smith, N. A. (2020), 'RealToxicityPrompts: Evaluating neural toxic degeneration in language models', in T. Cohn, Y. He and Y. Liu, eds, 'Findings of the Association for Computational Linguistics: EMNLP 2020', Association for Computational Linguistics, Online, 3356–3369.
- Gentry, C. (2009), A fully homomorphic encryption scheme, PhD thesis, Stanford University. crypto.stanford.edu/craig.
- Geva, M., Schuster, R., Berant, J. and Levy, O. (2021), 'Transformer feed-forward layers are key-value memories', in M.-F. Moens, X. Huang, L. Specia and S. W.-t. Yih, eds, 'Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing', Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 5484–5495.
- Ghorbel, A., Ghorbel, M. and Jmaiel, M. (2017), 'Privacy in cloud computing environments: a survey and research challenges', J. Supercomput. 73(6), 27632800. [Online], [Retrieved Jun 2025], https://doi.org/10.1007/s11227-016-1953-y
- Gunter, T., Wang, Z., Wang, C., Pang, R., Narayanan, A., Zhang, A., Zhang, B., Chen, C., Chiu, C.-C., Qiu, D., Gopinath, D., Yap, D. A., Yin, D., Nan, F., Weers, F., Yin, G., Huang, H., Wang, J., Lu, J., Peebles, J., Ye, K., Lee, M., Du, N., Chen, Q., Keunebroek, Q., Wiseman, S., Evans, S., Lei, T., Rathod, V., Kong, X., Du, X., Li, Y., Wang, Y., Gao, Y., Ahmed, Z., Xu, Z., Lu, Z., Rashid, A., Jose, A. M., Doane, A., Bencomo, A., Vanderby, A., Hansen, A., Jain, A., Anupama, A. M., Kamal, A., Wu, B., Brum, C., Maalouf, C., Erdenebileg, C., Dulhanty, C., Moritz, D., Kang, D., Jimenez, E., Ladd, E., Shi, F., Bai, F., Chu, F., Hohman, F., Kotek, H., Coleman, H. G., Li, J., Bigham, J., Cao, J., Lai, J., Cheung, J., Shan, J., Zhou, J., Li, J., Qin, J., Singh, K., Vega, K., Zou, K., Heckman, L., Gardiner, L., Bowler, M., Cordell, M., Cao, M., Hay, N., Shahdadpuri, N., Godwin, O., Dighe, P., Rachapudi, P., Tantawi, R., Frigg, R., Davarnia, S., Shah, S., Guha, S., Sirovica, S., Ma, S., Ma, S., Wang, S., Kim, S., Jayaram, S., Shankar, V., Paidi, V., Kumar, V., Wang, X., Zheng, X., Cheng, W., Shrager, Y., Ye, Y., Tanaka, Y., Guo, Y., Meng, Y., Luo, Z. T., Ouyang, Z., Aygar, A., Wan, A., Walkingshaw, A., Narayanan, A., Lin, A., Farooq, A., Ramerth, B., Reed, C., Bartels, C., Chaney, C., Riazati, D., Yang, E. L., Feldman, E., Hochstrasser, G., Seguin, G., Belousova, I., Pelemans, J., Yang, K., Vahid, K. A., Cao, L., Najibi, M., Zuliani, M., Horton, M., Cho, M., Bhendawade, N., Dong, P., Maj, P., Agrawal, P., Shan, Q., Fu, Q., Poston, R., Xu, S., Liu, S., Rao, S., Heeramun, T., Merth, T., Rayala, U., Cui, V., Sridhar, V. R., Zhang, W., Zhang, W., Wu, W., Zhou, X., Liu, X., Zhao, Y., Xia, Y., Ren, Z. and Ren, Z. (2024), 'Apple intelligence foundation language models'. [Online], [Retrieved Jun 2025], https://arxiv.org/abs/2407.21075
- Hou, X., Zhao, Y., Wang, S. and Wang, H. (2025), 'Model context protocol (mcp): Landscape, security threats, and future research directions'. [Online], [Retrieved Jun 2025], https://arxiv.org/abs/2503.23278
- Huang, J., Yang, D. and Potts, C. (2024), 'Demystifying verbatim memorization in large language models', in Y. Al-Onaizan, M. Bansal and Y.-N. Chen, eds, 'Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing', Association for Computational Linguistics, Miami, Florida, USA, 10711–10732.
- Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B. and Liu, T. (2025), 'A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions', ACM Trans. Inf. Syst. 43(2).
- Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A. and Fung, P. (2023), 'Survey of hallucination in natural language generation', ACM Comput. Surv. 55(12).
- Karamolegkou, A., Li, J., Zhou, L. and Søgaard, A. (2023), 'Copyright violations and large language models', in H. Bouamor, J. Pino and K. Bali, eds, 'Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing', Association

for Computational Linguistics, Singapore, 7403–7412.

- Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J., Zhang, H. and Stoica, I. (2023), 'Efficient memory management for large language model serving with pagedattention', in 'Proceedings of the 29th Symposium on Operating Systems Principles', SOSP '23, Association for Computing Machinery, New York, NY, USA, p. 611626.
- Lattner, C., Amini, M., Bondhugula, U., Cohen, A., Davis, A., Pienaar, J., Riddle, R., Shpeisman, T., Vasilache, N. and Zinenko, O. (2021), 'MLIR: Scaling compiler infrastructure for domain specific computation', in '2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)', 2–14.
- Lee, S., Shakir, A., Koenig, D. and Lipp, J. (2024), 'Open source strikes bread - new fluffy embeddings model'. https://huggingface.co/mixedbread-ai/mxbai-embed-large-v1. [Online], [Retrieved Jun 2025], https://www.mixedbread.ai/blog/mxbai-embed-large-v1
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S. and Kiela, D. (2020), 'Retrieval-augmented generation for knowledge-intensive nlp tasks', in 'Proceedings of the 34th International Conference on Neural Information Processing Systems', NIPS '20, Curran Associates Inc., Red Hook, NY, USA, 9459–9474.
- Mishra, M., Stallone, M., Zhang, G., Shen, Y., Prasad, A., Soria, A. M., Merler, M., ... and Panda, R. (2024), 'Granite code models: A family of open foundation models for code intelligence'. [Online], [Retrieved Jun 2025], https://arxiv.org/abs/2405.04324
- OpenAI (2024), 'Api reference - openai api'. [Online], [Retrieved Jun 2025], https://platform.openai.com/docs/api-reference/introduction
- Padhi, I., Nagireddy, M., Cornacchia, G., Chaudhury, S., Pedapati, T., Dognin, P., Murugesan, K., Miehling, E., Cooper, M. S., Fraser, K., Zizzo, G., Hameed, M. Z., Purcell, M., Desmond, M., Pan, Q., Ashktorab, Z., Vejsbjerg, I., Daly, E. M., Hind, M., Geyer, W., Rawat, A., Varshney, K. R. and Sattigeri, P. (2024), 'Granite guardian'. [Online], [Retrieved Jun 2025], https://arxiv.org/abs/2412.07724
- Pang, Q., Zhu, J., Möllering, H., Zheng, W. and Schneider, T. (2023), 'BOLT: Privacy-preserving, accurate and efficient inference for transformers', Cryptology ePrint Archive, Paper 2023/1893. [Online], [Retrieved Jun 2025], https://eprint.iacr.org/2023/1893
- Prastama, A. Y., Oliver, P. W., Saputra, M. I. and Titan (2023), 'User experience analysis of web-based application system otrs (open-source ticket request system) by using heuristic evaluation method', in S. C. Mukhopadhyay, S. N. A. Senanayake and P. C. Withana, eds, 'Innovative Technologies in Intelligent Systems and Industrial Applications', Springer Nature Switzerland, Cham, 487–497.
- Rho, D., Kim, T., Park, M., Kim, J. W., Chae, H., Ryu, E. K. and Cheon, J. H. (2025), 'Encryption-friendly llm architecture'. [Online], [Retrieved Jun 2025], https://arxiv.org/abs/2410.02486
- Sarwar, S. M. (2025), 'Fedmentalcare: Towards privacy-preserving fine-tuned llms to analyze mental health status using federated learning framework'. [Online], [Retrieved Jun 2025], https://arxiv.org/abs/2503.05786
- LM Studio (2024), 'Lm studio as a local llm api server'. [Online], [Retrieved Jun 2025], https://lmstudio.ai/docs/app/api
- Tabalba, Roderick S, J., Lee, C. J., Tran, G., Kirshenbaum, N. and Leigh, J. (2025), 'A pragmatics-based approach to proactive digital assistants for data exploration', in 'Proceedings of the 7th ACM Conference on Conversational User Interfaces', CUI '25, Association for Computing Machinery, New York, NY, USA. [Online], [Retrieved Jun 2025], https://doi.org/10.1145/3719160.3736632
- Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E. and Lample, G. (2023), 'LLaMA: Open and efficient foundation language models'. [Online], [Retrieved Jun 2025], https://arxiv.org/abs/2302.13971
- Voigt, S., Fuchs-Kittowski, F., Hüttemann, D., Klafft, M. and Gohr, A. (2011), 'Ickewiki: requirements and concepts for an enterprise wiki for smes', in 'Proceedings of the 7th International Symposium on Wikis and Open Collaboration', WikiSym '11, Association for Computing Machinery, New York, NY, USA, p. 144153. [Online],

[Retrieved Jun 2025], https://doi.org/10.1145/2038558.2038582

- Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q. and Rush, A. M. (2020), 'Transformers: State-of-the-art natural language processing', in 'Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations', Association for Computational Linguistics, Online, 38–45. [Online], [Retrieved Jun 2025], https://www.aclweb.org/anthology/2020.emnlp-demos.6
- Zhang, M., Press, O., Merrill, W., Liu, A. and Smith, N. A. (2024), 'How language model hallucinations can snowball', in 'Proceedings of the 41st International Conference on Machine Learning', ICML'24, JMLR.org. [Online], [Retrieved Jun 2025], https://proceedings.mlr.press/v235/zhang24ay.html
- Zhang, W. and Zhang, J. (2025), 'Hallucination mitigation for retrieval-augmented large language models: A review', Mathematics 13(5). 856. https://doi.org/10.3390/math13050856
- Zhao, S., Zhu, A., Mozannar, H., Sontag, D., Talwalkar, A. and Chen, V. (2025), CodingGenie: A Proactive LLM-Powered Programming Assistant, Association for Computing Machinery, New York, NY, USA, p. 11681172. [Online], [Retrieved Jun 2025], https://doi.org/10.1145/3696630.3728603
- Zhifei, X., Zongzheng, H., Zhang, G., Zhang, J., Liao, Y., Miao, C. and Yan, S. (2025), 'Pask: Providing answer before asking toward proactive ai agent', in 'Proceedings of the 33rd ACM International Conference on Multimedia', MM '25, Association for Computing Machinery, New York, NY, USA, p. 1355213554. [Online], [Retrieved Jun 2025], https://doi.org/10.1145/3746027.3754487
- Zuccon, G., Koopman, B. and Shaik, R. (2023), 'Chatgpt hallucinates when attributing answers', in 'Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region', SIGIR-AP '23, Association for Computing Machinery, New York, NY, USA, p. 4651.