ChatPLM Enhancements

Autumn McKee
Supervised by Nathan Soufflet

2025-08-29

Abstract

This report documents the enhancements made to ChatPLM, the chat-based AI assistant available from CERN's PLM platform. These enhancements included introducing robust validation and error-handling, extending the capabilities of the models used through Model Context Protocol (MCP) support, switching to an agentic pipeline to allow the model to autonomously continue to use tools to answer queries, and finishing the migration of the backend from Python to TypeScript. These changes, among others made, improve the resilience and utility of the system, and provide room for future enhancements.

1 Introduction

CERN's EN-IM-PLM section develops and maintains the Product Lifecycle Management (PLM) platform, which supports engineering efforts throughout the organisation[1]. In an effort to improve user support and reduce the burden on the support team, ChatPLM[2] [3] was developed as a chat-based AI assistant capable of answering user queries itself, using the already existing EDMS documents and how-to pages through semantic search.

The initial version of ChatPLM demonstrated the feasibility of using large language models (LLMs) for this task, however it left room for further improvements to be made.

2 Aims

The main objectives of the project were to:

- Replace linear pipeline with an agentic pipeline, enabling iterative tool use
- Extend model capabilities through Model Context Protocol (MCP) support
- Introduce robust validation and error-handling to reduce runtime errors
- Finish migration of backend from Python to TypeScript
- Improve document retrieval and SQL assistant

[1] https://proceedings.jacow.org/ipac2023/doi/jacow-ipac2023-thpa187/
[2] https://plm.cern.ch/chat
[3] https://gitlab.cern.ch/eis/plm/chat-plm

- Enhance the overall user experience

3 Implementation Details

3.1 SQL Assistant

The SQL assistant has been upgraded to be able to run read-only MS SQL queries, allowing the model to make queries to the database itself to answer user questions.

This change was done through adding support for Model Context Protocol[4], an open standard for providing tools to LLMs. This approach was chosen to avoid the burden of creating a custom adaptor, and with the aim of allowing other data sources to be added more easily in future.

Additionally, the system has been modified to provide all tools to the model simultaneously by default, allowing it to use as many data sources as available, however it remains possible to manually override which tools are used when needed.

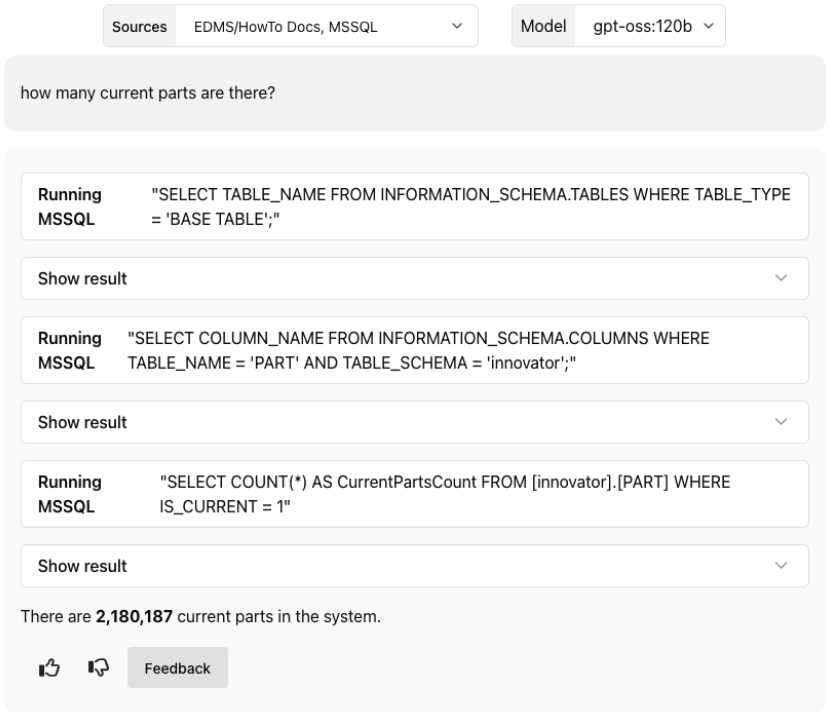

Figure 1: SQL Assistant workflow

3.2 Agentic Pipeline

Previously, the system used a linear pipeline for processing requests:

1. User asks a question
2. LLM calls a tool which returns more information
3. LLM responds using limited available context

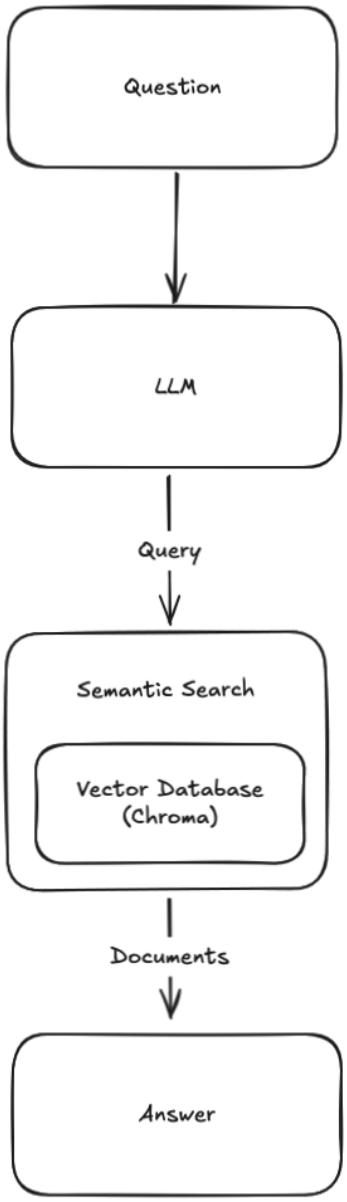

Figure 2: Linear pipeline

This limits the model to only answering using the information immediately available to it with a single tool call, which may not be enough to answer the user's query (e.g. complex multi-part queries).

Instead, the system now uses an agentic pipeline:

1. User asks a question
2. LLM calls a tool which returns more information
3. LLM evaluates if it has enough information to answer the question
4. If not, go back to step 2
5. LLM responds using wider available context

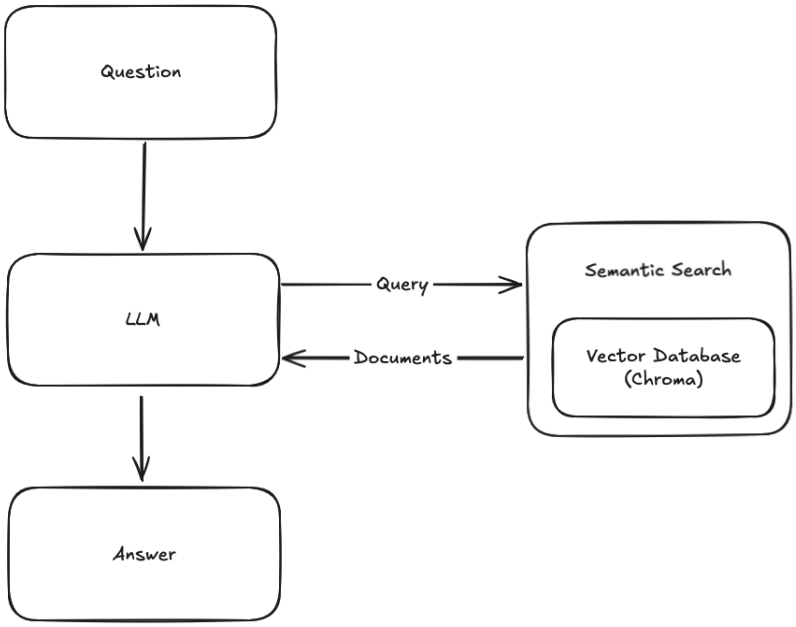

Figure 3: Agentic pipeline

3.3 Other Changes

A number of other changes/enhancements have also been made to the project, documented below.

3.3.1 Search and Retrieval

- Improved search/retrieval functionality when reranking service not available
- Improved fetching EDMS/HowTo docs, set up automatic re-fetching every night
- Allow the user to select any combination of information sources to provide to the model

[4] `https://modelcontextprotocol.io/`

3.3.2 User Experience

- Updated to allow streaming LLM response when tools are available, giving the user feedback sooner
- Support scrolling back to all previous conversations, instead of only limiting to the last 20
- Warn user if a conversation continues for too long, instead of silently erroring

3.3.3 Maintainability

- 2 years worth of dependency updates
- Allow system to be more easily extended in future by migrating answers in database to a new XML-based format
- Various other bugfixes

3.3.4 Reliability

- Reduce risk of errors at runtime through type checking with finishing migration of backend to Typescript, along with validation using Zod
- Improved XML parsing to be more reliable, by using a library instead of relying on a custom regex

4 Results

The aims of the project have successfully been achieved. As seen in Felix Lebel's evaluation of ChatPLM, the platform effectively gives accurate answers to user queries. Moving to an agentic pipeline has made it possible for ChatPLM to handle complex multi-part queries, especially in acting as an SQL assistant. The TypeScript migration has been fully completed and is now used in the production version of ChatPLM.

These enhancements also act as a solid foundation for future work, including adding additional data sources (e.g. Aras documents, support tickets), including Contextual Retrieval as a technique, and refinement based on further evaluation.