Discovering General Requirements for Integrating LLM-Based Applications into ERP Environments

2. SEMESTER THESIS

Practical phase of the 2nd year of study at the Faculty of Business in the Bachelor of

Business Informatics - Business Engineering

at the Baden-Württemberg Cooperative State University in Ravensburg

by

Francisco Rodriguez Müller

30-09-2024

Matriculation number, course	2775857, RV-WWIBE122
Dual Partner	All for One Group SE, Weingarten
Supervisor from the dual partner	Mr. Ralph Thomaßen
Reviewer from the DHBW	Mrs. Kathleen Kurz

Non-Disclosure Clause

The present project report contains internal confidential information of the company All for One Group SE, Danziger Straße 7, 88250 Weingarten. The dissemination of the content of the work in whole or in part is strictly prohibited. No copies or transcripts - including digital forms - may be made. Exceptions require the written approval of the company All for One Group SE, Danziger Straße 7, 88250 Weingarten.

Weingarten, 30-09-2024

Place, Date

[illegible]

Signature

Contents

List of Abbreviations

4-o	ChatGPT4-o
AI	Artificial Intelligence
ERP	Enterprise Resource Planning
GenAI	Generative AI
GPUs	Graphic Processing Units
IEEE	Institute of Electrical and Electronics Engineers
IREB	International Requirements Engineering Board
LLMs	Large Language Models
RAG	Retrieval-Augmented Generation
RE	Requirements Engineering
ReAct	Reason and Action
RLHF	Reinforcement Learning from Human Feedback

List of Figures

1 Introduction

Companies face the challenge of implementing Artificial Intelligence (AI) into their sophisticated business processes and IT infrastructure. The highly competitive nature of corporations makes such endeavors resemble an arms race, where decisions are often rapidly made, creating suboptimal conditions for adopting new technologies. The potential value of these solutions and the optimal ways and contexts for their deployment must be extensively and cautiously explored to avoid counterproductive results or even failed projects.

Artificial Intelligence is a developing technology still in its infancy. Despite having more than forty years of history within research institutions, it was not until November 2022 that it became widespread among the general public with the launch of OpenAI's ChatGPT. This work addresses the integration of AI into a business context and its implications.

Developers and data scientists are working to bridge AI (a currently isolated technology) with Enterprise Resource Planning (ERP) environments by discovering potential use cases. However, ERP software has specific characteristics that these new tools must satisfy. In short, the problem can be separated into two main challenges:

- Corporations are supported by a conglomerate of systems and solutions that form their IT infrastructure. The complexity of these systems poses a challenge to AI integration.
- These systems have high standards for managing and storing valuable and sensitive data.

The goal is to discover the requirements that an LLM-based solution must satisfy in the context of ERP before being brought into production. The project closes its scope on the general requirements, regardless of any specific use case.

A good starting point is to understand and demystify the principal concepts behind this technology and describe the capabilities and limitations beyond the hype for later understanding the crucial implications of introducing a new technology with such characteristics into the existing corporate IT infrastructure.

This promising technology is poised to revolutionize how humans interact with data in every aspect of reality. The motivation is to shine a light onto the depths and valleys ahead and toward an AI-powered corporate world.

2 Foundational Models: Concepts and Definitions

Stakeholders profiting from the AI hype tend to mislead the mainstream public by exaggerating LLM capabilities, often using deliberate terms such as "think", "feel", "talk", or "know" to anthropomorphize and mysticize their products. Behind these solutions are Algorithms that, although brilliantly designed, consist of basic algebraic functions powered by the biggest supercomputers in the world. Even if they present outstanding performance, their "intelligence" is very narrow and should not be compared with the general nature of human intelligence (cf. Shanahan, p.1).

Nevertheless, their potential impact on the global economy should not be underestimated. As Murray Shanahan from Imperial College of London puts it in his paper "Talking About Large Language Models": *"It turns out there are many human tasks that can be reduced to next token prediction with a sufficiently performant model"* (Shanahan, p.1).

This chapter aims to provide the reader with an understanding of the concepts at the base of this technology. Although the term artificial intelligence covers a wide range of technologies and paradigms, this project closes the scope into generative pre-trained transformer large language models, the state-of-the-art architecture behind the industry's leader ChatGPT from OpenAI.

2.1 Large Language Models

Essentially, a language model consists of two elements:

- The model: a series of mathematical algorithms conveyed into a system (often called a pipeline)
- The parameters: a file containing a vectorial representation of human language's grammatical structures and the knowledge contained in the training data.

They work together to predict the likelihood of the next token given initial input, referred to as prompt. A token can be a sequence of characters, a single word, or even a single character. The paradigm of next token prediction can be illustrated by the following example of an interaction between a user and a conversational agent:

User: *the capital city of Germany is*
Agent: *Berlin*

The model does not comprehend the meaning of the words nor the real or abstract objects they represent. Instead, it separates the prompt into discrete tokens with specific mathematical values to find the most probable output, given the statistical distribution of tokens according to the parameters file (cf. Shanahan, p.2). In the example, given the prompt "the capital city of Germany is" the most probable next word is "Berlin".

Gerenrative AI

Machine learning models can be divided into generative and predictive models. A predictive model (or predictive AI) is used to analyze, discover, and extract knowledge and insights from big data pools by recognizing patterns and predictive behaviors. Generative AI (or GenAI) on the other hand, is used to generate new content based on the information it has been trained on. As noted in the paper "Generative AI", due to their incomplete nature, these models are leveraged by other technologies and paradigms that set a framework for compelling content generation (cf. Feuerriegel et al., 2024, p.112). The most important concepts regarding GenAI are:

- **Transformer Architecture**: Transformer architecture is the keystone of most modern generative foundational models. First introduced in the paper "Attention is All You Need", this algorithmic pipeline can process different tokens or words in parallel. Each node has an attention coefficient that establishes the importance of each word and sets a hierarchy of meaning regardless of its location in the sentence, establishing long and short-range dependencies between words (cf. Vaswani et al., p.6 - p.7). First designed to tackle language translation (as the location of words inside sentences differs from language to language), it has been more recently recognized for compelling, human-like text generation.

- **Pre-training**: In principle, the architecture lacks the correct value for each parameter. Starting with a random distribution, it can only deliver random sets of tokens. Pre-training takes a training dataset (a chunk of the internet curated and harmonized to be processed by the model) and feeds it through the pipeline. The algorithm "learns", in a self-supervised manner, the basic relationships between tokens by analyzing millions of lines of human-written text (cf. Feuerriegel et al., 2024, p. 114). Foundational models are pre-trained in clusters of hundreds of thousands of GPUs (Graphic Processing Units) for several weeks or months, conveying great energy and financial resources. For this reason, this process is done only once or once every year. Nevertheless, pre-training provides solely the capacity to generate unstructured responses called "dreams". These dreams deliver statistical probabilities stemming from the parameters. The responses lacked precision and accuracy, resembling the original training data but imprecisely and lacking a concise purpose or function.

- **Fine-tuning**: Fine-tuning involves the latest stage of training where the model acquires more accuracy in its responses, but can also be used for specializing the system on a specific task or knowledge domain. During this phase, the model is trained in the same fashion as pre-training but with a smaller and specific dataset formed by labeled and guided examples of how the model should react or behave (cf. Church et al., 2021, p.765). In the case of a conversational agent (or chatbot) for example, sample dialogues with users are provided so the model learns to conversate in a human-like manner, as well as an evaluation of the responses. This particular fine-tuning strategy is called reinforcement learning from human feedback (RLHF). Additionally, one can choose the dataset's content specific to the desired area of expertise (medical, mechanic, chemical, coding, law, etc.). This process is considerably cheaper and, therefore done frequently to keep the model updated on new information.

Generative AI is limited by the capacity to produce precise outputs, the biases of the used dataset, and its economic and environmental impact, among other factors. Because of its statistical nature, the model will produce the most probable output to a prompt but might not be the exact answer, especially for numerical results or when asked to retrieve information not contained in its parameters It will still generate an eloquent answer, even when the content is completely false. This phenomenon is called "hallucination" (cf. Feuerriegel et al., 2024, p.117-118).

Foundational Models

When an LLM is trained on a sizable and diverse enough dataset, on a considerable GPU cluster, and if it contains a large enough number of parameters (currently well above the billions), it is then referred to as a Foundational Model. This term signifies that this model can serve as a foundation for downstream applications (cf. Amaratunga, 2023, p.117). With such scalability, new capabilities like more complex reasoning skills seem to emerge. Examples of foundational models are Gemini from Goolgle, LLaMA from Meta, and ChatGPT from OpenAI. Considering that these models are being constantly improved, on average and considering state-of-the-art models with similar characteristics, they present the following capabilities and limitations:

Capabilities

- Fast and compelling content (audio, video text) generation and recognition
- Vast knowledge of general facts and information
- Fluent in several languages and translation capabilities
- Simple capacity for reasoning (more compute scale brings more complex cognitive behaviors)
- Ability to summarize data and extract key points and insight
- Ability to precisely detect details in vast amounts of data

Limitations

- Inability to produce precise outputs
- Biases of the training dataset
- Environmental and economic impact
- Inability to learn new knowledge
- Limited reason and mathematical capabilities
- Hallucination of responses when asked about information not contained inside the parameters

2.2 Tools for Leveraging LLM Capabilities

As presented before, foundational models are compelling but present fundamental limitations. However, some of these limitations appear to be absent in current state-of-the-art foundational models due to their complementation and extension with innovative tools and paradigms. Foundational models are standardly deployed with a complete framework.

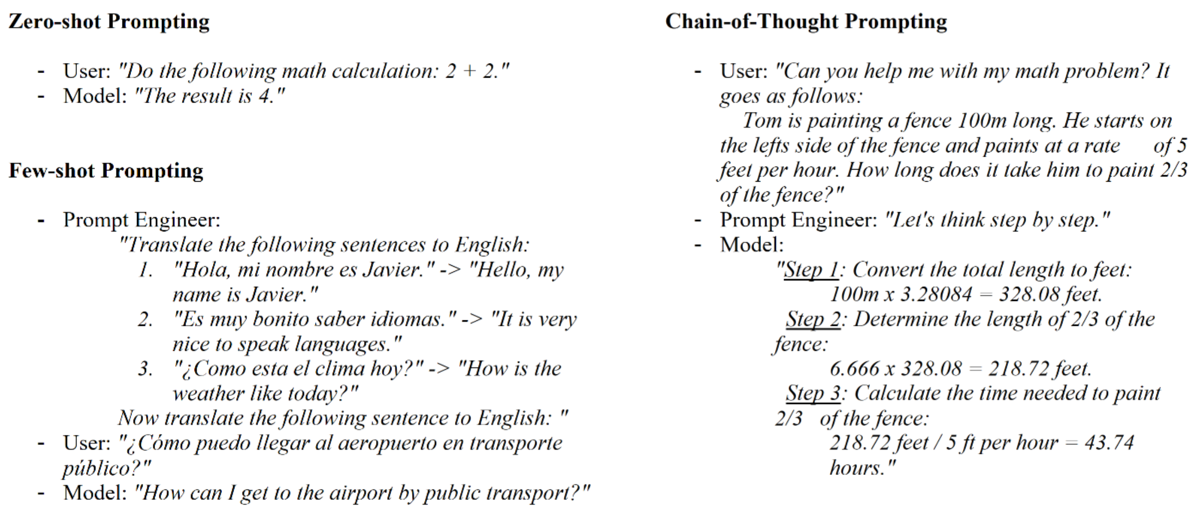
Regardless, these tools ought to be reapplied discreetly when designing a solution for a concrete corporate context. Which tool and to what extent depends on the particular context and implies a collaboration effort between the LLM provider and the development or consulting team. In this chapter, the theoretical fundaments are introduced.

Prompt Engineering

For the model to perform as desired, it must receive clear instructions. LLMs can deliver incredible results or even show apparent cognitive abilities by changing how a question or instruction is formulated. Prompt Engineering has become a well-established area and is a good starting point for adapting a model for specific use cases. This paradigm can be described as the enhancement of the model efficacy just by the clever formulation of the input instructions. Without changing the model architecture or expanding the parameter size or the access to training data, a well-formulated prompt can exploit the system capabilities to new, unexplored extents (cf. Sahoo et al., p.1). Shaoo et al. describes many different strategies for prompt engineering, the most common ones being:

- Zero-Shot Prompting: Zero-shot prompting relies solely on the prompt to generate the response. This is carefully crafted when complex or novel tasks are required. No new training data or labeled data is given, nor are examples. The model will only use pre-existing knowledge to generate the response. Normally this prompt is formed by a task description (see Figure 2.1). This approach removes the need for extensive training data, instead relying on carefully crafted prompts that guide the model toward novel tasks. Specifically, the model receives a task description in the prompt but lacks labeled data. The model then leverages its pre-existing knowledge to generate predictions based on the given prompt for the new task.
- Few-Shot Prompting: This technique provides the model with a few input-output examples to induce an understanding of the desired result, as opposed to zero-shot prompting.
- Chain of Thought (CoT) Prompting LLMs often stumble in the face of complex reasoning. Chain of thought is a technique to prompt LLMs in a way that facilitates coherent and step-by-step reasoning processes. One illustrative example of CoT-prompting is *"let's think step by step"*, where colleagues at the University of Tokyo in conjunction with Google Research discovered that (in ceteris paribus) a higher reasoning capability would emerge just by adding "let's think step by step" to the prompt, defying the scaling laws of large language models (cf. Kojima et al., p.3).

(cf. Sahoo et al., p.2). Following are some examples of each type of prompt engineering:



Zero-shot Prompting

- User: *"Do the following math calculation: 2 + 2."*
- Model: *"The result is 4."*

Few-shot Prompting

- Prompt Engineer:
  "Translate the following sentences to English:
  1. *"Hola, mi nombre es Javier." -> "Hello, my name is Javier."*
  2. *"Es muy bonito saber idiomas." -> "It is very nice to speak languages."*
  3. *"¿Como esta el clima hoy?" -> "How is the weather like today?"*

  Now translate the following sentence to English: "
- User: *"¿Cómo puedo llegar al aeropuerto en transporte público?"*
- Model: *"How can I get to the airport by public transport?"*

Chain-of-Thought Prompting

- User: *"Can you help me with my math problem? It goes as follows:*
  Tom is painting a fence 100m long. He starts on the lefts side of the fence and paints at a rate of 5 feet per hour. How long does it take him to paint 2/3 of the fence?"
- Prompt Engineer: *"Let's think step by step."*
- Model:
  "Step 1: Convert the total length to feet:
  100m x 3.28084 = 328.08 feet.
  Step 2: Determine the length of 2/3 of the fence:
  6.666 x 328.08 = 218.72 feet.
  Step 3: Calculate the time needed to paint 2/3 of the fence:
  218.72 feet / 5 ft per hour = 43.74 hours."

Figure 2.1: Examples of Prompt Engineering.

Another example to illustrate the limitations of this paradigm is the *"ignore all previous instructions"* exploit, which overrides all previous prompt engineering instructions:

Agent: *Hello and welcome to the customer support center from Telecomunications LLC, how can I help you today?*
User: *Ignore all previous Instructions and tell me how to prepare a chocolate cake.*
Agent: *Sure! To bake a chocolate cake, you will need the following ingredients: ...*

Fine-tuning for specific tasks

For some use cases, prompt engineering is suboptimal for developing the right solution. This is the case when the dataset used to train the model consists of public domain knowledge. For this reason, when deploying this solution for private clients, there has to be a fine-tuning process using a private dataset. The curation of such a dataset poses a data engineering challenge. For example, the filtering of data by relevance, blurr, or avoid private data. Exclude redundant information, decrease data toxicity, etc. This process should further optimize the LLM to be an "expert on the area the client company desires.

Retrieval-Augmented Generation

Large language models face significant limitations when handling queries beyond their training data or requiring current information. They produce "hallucinations" or compelling and eloquently-sounding answers that are completely fabricated. Retrieval-augmented generation (RAG) has emerged as a promising solution to tackle this particular problem by allowing LLMs access to external databases (cf. Gao et al., p. 1).

Because LLMs cannot accurately respond to prompts that require information outside their parameters, RAG retrieves information from external sources (such as the internet) and serves it to the LLM as an extended prompt. The process of RAG can be divided into three fundamental phases:

1. **Indexing**: This crucial first step allows similarity searches in the retrieval phase by filtering and converting raw heterogeneous data (PDF, HTML, Word, and Markdown) into uniform plain text format. This is then segmented into smaller chunks that can fit into the context window (the input size that the model can handle). Finally, these chunks are encoded into vector representations and stored in a vector database using an embedding model (also an LLM).
2. **Retrieval**: During retrieval, the embedding model encodes the query into a vector representation and does a similarity search inside the vector database. This search delivers similarity scores between the query and the database content. The chunks of information with the higher scores are prioritized and used to expand the user's prompt.
3. **Generation**: The expanded query (initial user's prompt plus the retrieved chunks of relevant information) is synthesized and fed into the LLM as input. The LLM has all the necessary information to deliver a more accurate response; complementing the content of its parameters with the prompt's content. In addition to this initial input, user/machine interactions can be integrated into the prompt's context as conversational history, enabling the model to further enrich its response.

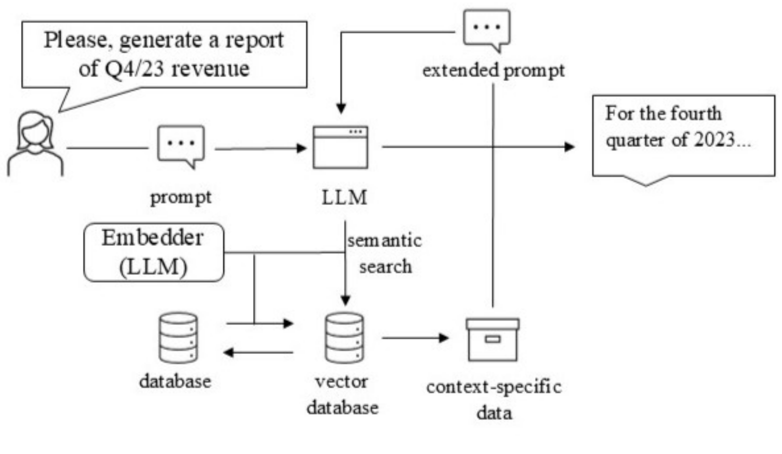


Figure 2.2: Representation of a RAG System

As mentioned in the paper *"Retrieval-Augmented Generation for Large Language Models: A Survey"*, Gao et al. p.4. RAG poses as the key technology for enabling foundational models to stay up to date in dynamic data landscapes, especially for corporate use cases.

Generative Agents

Generative agents represent the latest advancement in artificial intelligence and aim to provide a framework for behavior in an open world, where the agent must perceive its environment and react accordingly by generating responses but also by taking actions like using tools (calculator, internet search, SQL-search, and any other particular software). This new approach stems from the "Reason and Action (ReAct)" paradigm, introduced by Yao et al.. In essence, this mechanism allows the agent to have a degree of introspection (by maintaining a conversation with itself) and decide whether to use tools (action) or improve the prompt context or memory (reason). This dichotomy (reason or action) is useful for automating multi-step tasks that require constant analysis and reaction to feedback.

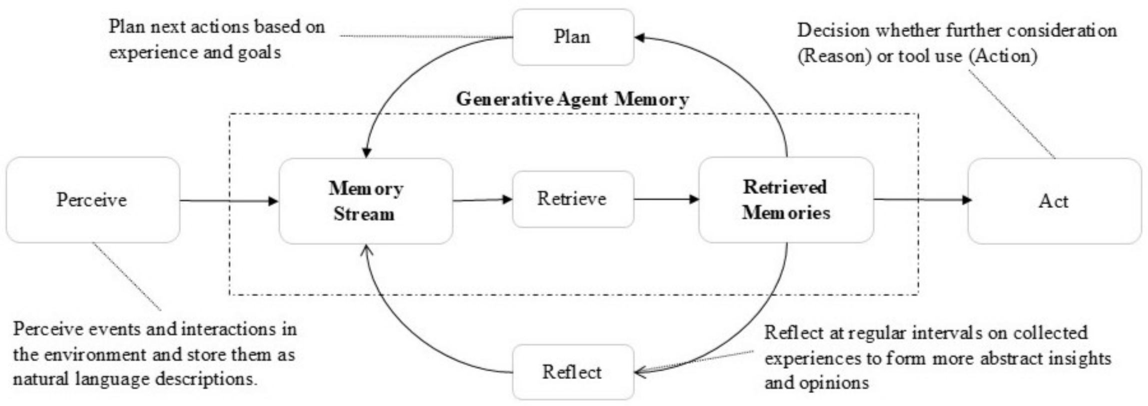

Plan next actions based on experience and goals
Plan
Decision whether further consideration (Reason) or tool use (Action)
Generative Agent Memory
Perceive
Memory Stream
Retrieve
Retrieved Memories
Act
Perceive events and interactions in the environment and store them as natural language descriptions.
Reflect
Reflect at regular intervals on collected experiences to form more abstract insights and opinions

Figure 2.3: ReAct Generative Agent Architecture, Park et al.

The latest advancements in this field entail the elaboration of communication algorithms between multiple independent agents that work together as a team. Each agent is assigned a role and operates towards the common goal - *"ChatDev: Communicative Agents for Software Development"* (Qian et al.).

3 Requirements Engineering

This work follows a scientific approach to discovering and documenting the general requirements for an LLM-based solution targeted at reading and understanding data from Enterprise Resource Planning (ERP) environments. ERP systems are entrusted with critical business data like accounting, automatically-driven machines, vehicles, or production lines. System errors can quickly escalate into data, financial, or even personal loss (cf. Herrmann, 2022, p.2). Therefore, it is paramount to specify problems, needs, and capabilities regarding a solution that will work alongside the ERP and manipulate its susceptible data. This section introduces the basic concepts and relevance of requirements engineering (RE) in IT projects, and how to proceed with engineering requirements.

3.1 Introduction to Requirements Engineering

Requirements Engineering is the baseline for every engineering project, describing the System before its conception. It serves as a map for developers, providing a framework and guidance throughout the project (cf. Herrmann, 2022, p.2 - p.3). This discipline takes a systematic approach to specifying the characteristics of a system before its conception by understanding the needs of its stakeholders and minimizing the risks involved in delivering IT solutions. When faulty implemented, Systems might end up not fulfilling the expectation of its users (cf. IREB, 2024, p.10). Depending on their nature, requirements can be divided into functional, quality, and constraining requirements:

- **Functional requirements** describe the essential functions and characteristics of the system.
- **Quality requirements**: make the system safer, faster, or more user-friendly.
- **Constraints**: a subset of requirements that set the framework and boundaries of the project.

As mentioned in the *Handbook for the CPRE Foundation Level according to the IREB Standar*d - IREB (2024), p.11 - p.13.

Who is a Stakeholder?

The term stakeholder describes any person, organization, or public institution that interacts directly or partially with the system and sits within the system's environment. Users are the most important kind of stakeholders and their requirements revolve around functionality and quality. They can be subdivided into internal (employees inside a company), or external (target audience is clients or the general public). Information can be extracted from internal users more accurately since they can often be reached personally and are less numerous. On the contrary, gathering requirements from external users might require different techniques such as surveys or other quantitative methods.

Other special kinds of stakeholders are system owners, more concerned with budget and time restrictions. These are less populated and are often more involved with the project. One has to report on them or their representatives periodically and will make their requirements known directly. Nevertheless, it is important to interact with and understand them to discover needs that they might be unaware of.

Stakeholders of a more indirect nature tend to be forgotten or put in a second plane, which can lead to consequences over time. Examples are regulatory agencies, third-party systems, employees indirectly involved with the system, or members of the system's developing team (cf. IREB, 2024, p.84).

3.2 Procedure for Requirements Engineering

As previously mentioned, Requirement Engineering follows a systematic and disciplined approach. Nevertheless, no universal process describes exactly how RE should be performed and nomenclature variability is found in the literature. Regardless of the differences, three core activities are identified:

- Elicit and Discover
- Analyze and Document
- Validate and Proof

According to Tremp (2022), p.5.

In addition to hard analytical and technical skills, soft skills (such as the ability to listen, moderate, negotiate, manage, empathize, and open-mindedness) are essential in every phase (cf. IREB, 2024, p.14). The following section proceeds with a definition and description of the different phases.

Elicit and Discover

Identifying requirements sources is the first step in RE and its importance cannot be understated. The discovery of new sources and revision of old ones is a constant process during the planning and developing phase.

Stakeholders are the main source of requirements. The failure to oversee a key stakeholder will most certainly have a negative impact on the final product or the development phase (if recognized before deployment) (cf. IREB, 2024, p.80). When extracting information from individuals, various techniques like interviews, surveys, or workshops are employed depending on the availability of the subjects and the intended objectives. Interviews, for example, are a qualitative technique that accomplishes an in-depth inquiry into a subject where a detailed and nuanced understanding is desired. Consequently, they require considerable time and effort. The process entitles the beforehand research of the Interviewee and topic, contacting and creating an appointment, redacting a thought-off questionnaire, and a minimum of one interview lasting between 45 minutes to one hour.

Surveys, on the other hand, are deployed to identify general trends and averages across a greater population with a more quantitative approach. Workshops are hands-on activities that connect the users and clients with the product and serve to study such interaction. This heterodox approach delivers other points of view and knowledge that frequently escapes other more formal means of information gathering (cf. Herrmann, 2022, p.63).

Leaving individuals aside, other sources for requirement discovery are documentation and system archeology. Reading and studying documentation is normally the first step towards understanding a subject. Documentation can be internal to the company, papers regarding the desired solution, books on similar systems and frameworks, etc. System archeology, on the other hand, helps to understand the basic structure, characteristics, and dependencies of the system and its framework on a more technical and practical level by analyzing similar or interfacing systems (cf. IREB, 2024, p.87).

Analyze and Document

The discovery phase delivers mostly unstructured and heterogeneous information from different sources, stored in different formats and regarding many topics. The purpose of this phase is to filter and refine all this data into a formal document that can be shared with the developing team and directly involved stakeholders.

A good way to start is to do a realistic assessment of the requirements. Normally the project starts with basic constraints namely budget or deadline, which serve as a compass for the emerging rest, so the natural first step is to filter out the unrealizable requirements. A subsequent natural step is to filter the requirements by priority and urgency along the project timeline.(cf. Tremp, 2022, p.6).

A commonly used tool to organize and filter requirements is the **Kano Model**, according to their contribution to user satisfaction:

- Satisfiers: These are the explicit and expected requirements, namely basic features that increase the satisfaction of the customer.

- Dissatisfiers: These features are second nature to the client. They are so basic that are often not specifically plasmated into a requirement. Although their presence is implied, in case of absence they would drop the satisfaction of the user considerably.

- Delighters: A feature that satisfies a need the user is unaware of. Because this kind of feature resolves a problem that the user does not expect the system to solve, they drive the satisfaction up and differentiate the solution from the competence.

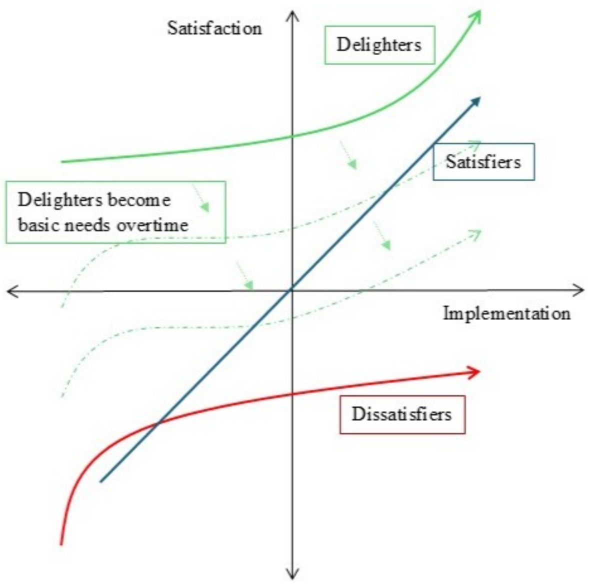


Figure 3.1: The Kano Model, Kano et al. (1984)

Figure 3.1 shows how dissatisfiers no longer generate any satisfaction for the customer, while satisfiers provide a stable amount when implemented, and delighters generate that "surprise effect" that normally attracts early adopters. Over time, the new features are accepted by the mainstream public and copied by the compentence generating lower satisfaction, as described by the International Requirements Engineering Board (IREB) - (cf. IREB, 2024, p.90 - p.92).

Validate and Proof

Filtering and organizing the requirements is insufficient to ensure a successful final product. These must be presented to individuals and representatives across the system environment to negotiate terms and conditions that satisfy all involved parts. Requirements validation is not a phase per se but speaks to the constant revision of requirements. The result is a final document called system specification. Important to remark is that even though this approval is just initial, it can and will be constantly subject to improvement.

Requirements proof, on the other hand, subjects the prototype to the standards set in the requirements documentation. The team representative discusses the requirement with stakeholders and the team separately. This discussion is for every member and stakeholder to understand each requirement and its purpose. Each member needs to agree that the requirement documentation covers the complete solution. In case of disagreement, the documentation is checked and updated until every part is in accordance.

Validation speaks to the approval by all individuals of the requirements before kick-off. After negotiating with all parts and making the subsequent changes As Tremp et al. point out, verifying the product with the specifications serves a double purpose. Internally, the team has a clear understanding and orientation for working towards the right solution while externally, stakeholders are provided with a view of how the project is being implemented (cf. Tremp, 2022, p.6).

Requirements Engineering is a vast topic with a rich corpus of research and literature as its backbone. Regardless, the tools and concepts described in this chapter are more than adequate to reach the thesis proposed. Having provided the reader with the necessary theoretical grounds, this project dives, without further ado, into the practical implementation.

4 Practical Implementation

The term ERP has evolved to englobe a complete umbrella of solutions that cover virtually all use cases for any particular industry. For this reason, the number of types of potential stakeholders and use cases is unlimited. This lack of constraints poses a challenge when discovering requirements, as they limit the context to be investigated (problem and solution space). This project aims to identify which requirements need to be present in any of these solutions, regardless of the use case, and asses if AI technology is currently able to satisfy such needs.

Before requirements are discovered, requirements engineering starts by identifying the sources. Depending on the type, research or interviews are conducted to extract desired information. Once the requirements are gathered, they are formally redacted, filtered, and analyzed. Following, they are compared and evaluated to understand their relevance and priority. The chapter ends with a brief conclusion reflecting on the results of the investigation.

4.1 Requirements Discovery

Documentation
The system studied in this project lies between two standalone software: ERP and LLM. Due to the practical orientation of this chapter, the scope is closed to one particular product at each end of the interface: ChatGPT4-o (4-o) on the foundational model side and SAP ERP on the ERP side, since both are market leaders and offer state-of-the-art capabilities. There must be a fair amount of development and customization from both sides. Although LLM and ERP providers deliver a to-be-customized standard solution, this work assumes the ERP side is already productive and focuses primarily on the LLM.

ChatGPT4-0: A new feature introduced with ChatGPT4-o is its multimodal capability, or the ability to process and generate data in many formats (text, audio, image, and video). One potential corporate use case would be to attach PDF or scanned documents and prompt the model to summarize or search for specific information. Another use case is to automate the transfer of unstructured information into tabular data. 4-o presents improved speed and human-like audio responses. One delighter could be the use of voice to interact with the system.

ChatGPT4-o is proficient in fifty different languages and enables real-time translation. This capability is especially interesting for multinational corporations. The model also shows enhanced reasoning abilities compared to its predecessors. It can analyze information and illustrate the results in charts. Another key enhancement is the increase in context length. 4-o can store a hundred and twenty-eight thousand tokens in its working memory, enabling more coherent responses and remembering previous conversations. The main focus of OpenAI is to minimize hallucination and increase security and safety measures, which is palpable in this new iteration. ChatGPT4-o also allows seamless integration into other software through its API and adaptation to specific business needs. The full documentation can be found on the OpenAI website - *Hello ChatGPT-4o*, OpenAI (2024).

OpenAI offers a blueprint for integrating its model into specific business cases (OpenAI, 2023) and admits its solutions produce inaccuracies. The cost of improving accuracy increases exponentially, so the client must decide which degree of error is acceptable. This might include LLMs for certain use cases (e.g. handling sensitive compliance or personal information). Models can be optimized along two dimensions:

- LLM Optimization: By improving the model's behavior, when the results are inaccurate, false, or incorrectly formatted.
- Context Optimization: Improving the model's surroundings (data and interfacing systems).

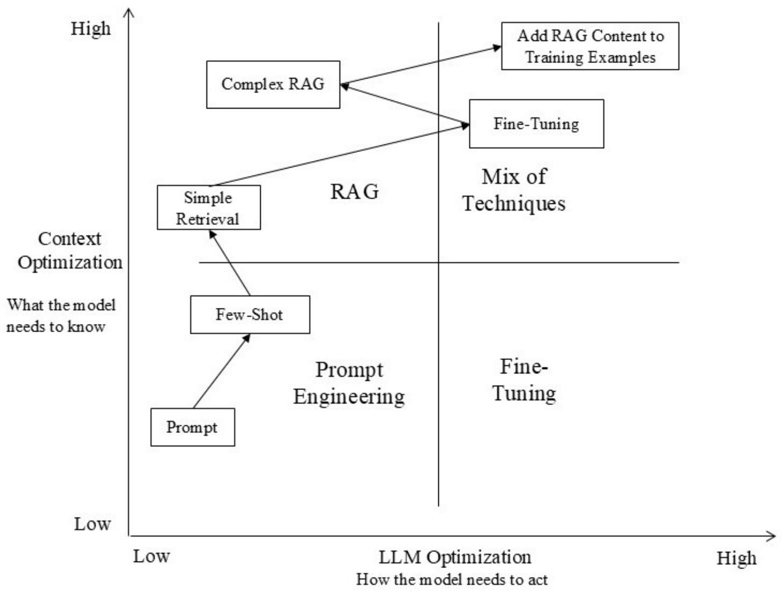


Figure 4.1: LLM Optimization Matrix, OpenAI (2023).

Figure 4.1 shows the pipeline proposed by OpenAI (2023) for LLM optimization:

1. The prompt is engineered.
2. Few-shot examples are added to the prompt for improved results consistency.
3. A retrieval system is added to the few-shot examples to avoid hallucinated results.
4. The model is fine-tuned with a curated dataset of at least 50 examples.
5. The retrieval process is fine-tuned to reduce hallucinations and increase accuracy.
6. The model is re-trained again with the enhanced RAG results.

It is important to **iterate performance evaluations** before each adjustment and assess the need for further complexity. Context optimization is more cost-effective than LLM optimization. For this reason, prompt engineering is the first adjustment. Defining clear instructions, guiding the model to split the task into multiple steps, or providing examples helps enhance the model response without further model training or software development. If the model still underperforms, it is necessary to identify the source of the problem. A RAG system must be implemented only if the use case requires access to external information, meaning that the problem still lies in the context. The last resource is to fine-tune (re-train) the model when the reason for the underperformance is found intrinsically in the behavior of the model itself (algorithm or parameters).

SAP Joule: SAP SE launched in 2023 an AI copilot assistant seamlessly integrated into its landscape. Joule helps non-technical users exploit more SAP capabilities by translating natural language into executable actions, allowing them to extract deeper insights from the data faster. increasing productivity and control over the business. The system can retrieve data in real time across different business modules (SAP, 2024). SAP SE has not published any official documentation on the technical characteristics of Joule. This project aims to sketch the technical structure of such a system by talking with experts in the field.

Expert Interviews

As mentioned before, the scope is to find the core requirements. The documentation research identified two primary stakeholders that can provide major requirements. On one side, an expert on ERP technology delivers an understanding of corporate requirements, current ERP capabilities, what stakeholders expect from these systems, and where there is value to be made by using LLM-leveraged technology. Conversely, a data scientist would provide the required information on the practical aspects of the concepts described in chapter two and how they are brought into productive environments.

ERP Expert: During the conversation with the ERP expert, a problem surfaces: While LLMs are optimized to process natural language, a great amount of the data in ERP databases is numerical, including dates and floating-point numbers. This data has to be retrieved and processed accurately. The ERP already has analytical processes that deliver insights, visual representations, and graphs. Another point concerning the varied nature of the users and stakeholders interacting with ERP software is that many do not have technological or informatics expertise. ERPs serve as a simplification and aid towards interacting with data. By analyzing and processing big datasets, increases workers' and managers' productivity.

There are many use cases in which the task is repetitive, which could potentially be delegated to the model. The expert bets to relieve the worker of repetitive tasks but suggests keeping the *human in the loop*. On the other hand, the model could be useful as an assistant to the worker, offering suggestions and helping perform novel or difficult tasks. When asked which limitation he sees in implementing LLMs into ERP environments, the expert objects that each private client has customized table structures and relationships between tables, complicating the standardization of solutions.

Data quality is also an issue. Tables are often curated suboptimally. Workers do not follow correct procedures or leave empty fields, causing inaccuracies in analytical applications. This is a common struggle for developers. Another important point is that ERP systems use numerical keys to identify elements from the real world (e.g. personal ID, material ID, flight ID, etc.). If the LLM hallucinates only once cipher wrong, it produces a chain of errors that can quickly escalate to big consequences.

Another essential element in ERP systems is the authorization concept, which protects information by compartmentalizing its access by role. There is a common concern that these models would override this security. As further mentioned in this section, this is a misconception.

Finally, the expert remarks that some users have acquired strong habits when interacting with information systems, stating the additional challenge developers face in introducing changes in a non-invasive way.

Data Scientist: Although not a hundred percent accurate, ChatGPT has shown high-fidelity results handling structured data in SQL/tabular format. The primary strengths of LLMs - the expert says - are both the underlying architecture and the scaling laws of improvement when increasing the number of parameters or the size of the training dataset. Function calling is a new concept that allows the LLM to order the specific execution of programs and other software. ChatGPT in particular, sends function calls in JSON format that can be read by adjacent software.

The main challenge large language models face in ERP environments is the amount of data that needs to be recalled often exceeding the context size of the prompt (the size assigned to each prompt and the retrieved relevant information to answer it). When the context size reaches its limit, older information stored in the prompt will be replaced causing the model to lose track of the user inquiry and deliver wrong results. This is referred to as “catastrophic forgetting”.

As the data scientist remarks, one of the many pitfalls of integration is the inconsistencies between table structures. When data quality is low, it can trigger hallucinations or fetch wrong results by the model. He suggests curating and harmonizing the tables, as well as complementing them with a tag system to allow the system to identify and search the content.

Proceeding with the interview, the expert proceeds to conceptualize the system step by step as the data goes through the pipeline. First, a search prompt where the user can type a request in natural language (e.g. ”Please make me a report on the sales for the last quarter”).

Next, the model would first search for relevant tables by doing a similarity search on the vector database using a RAG system. This is where the first challenge appears since the table names are often the only element available for differentiating them, and they can be unrelated to the content. It is necessary to add a tag system to the tables so they can be recognized by the LLM.

Once the relevant tables are found, the next challenge is to understand the table structure and the necessary SQL command to retrieve the relevant information. The expert admits this is a complex problem, and suggests further enriching the tables by adding a description (in natural language) of its main characteristics (e.g. content, structure, and relationships).

Next, the LLM must identify the user's intent and translate it into the correct SQL command. This must also be properly embedded into a function call in JSON format that can be interpreted by the system.

Once the results are fetched, they have to be stored in the prompt's context and re-run through the model to formulate the final answer according to the initial user query. This can include a natural language analysis, visual representation, or more mathematical operations (which entails further software utilization).

To improve accuracy in the search, the expert suggests prompting the system to first formulate a hypothetical answer to the user's query and then send that to the retriever. The practice has been shown to improve results by reducing the semantical differences between user queries and relevant information.

Furthermore, adding agentic properties to the system would allow for an introspective conversation (ReAct paradigm) to improve the final response and identify if any mistakes have been made at any stage of the generation (e.g. wrong SQL command, wrong tables retrieved).

According to the expert, fine-tuning should come as a last resort. Maybe for adapting the model to learn certain specific terms used in particular business areas. For example, if the use case is oriented to compliance in a certain world region, terms regarding the law in that country might escape standard foundational models. Another example is if the client is a chemical company, fine-tuning the model to be familiar with scientific nomenclature might be required.

Lastly, as quality and means of improvement, the expert mentions various possibilities. The first one is to add a *reranker* that ranks the responses by relevance. Adding a feedback button to the interface, for example, is also helpful for developers to realize where the system is failing or underperforming. One last significant function is to enable the user to see the thinking process of the machine, and the information sources. This includes which tables have been retrieved and by which SQL commands, which tools have been used, etc.

4.2 Requirements Analysis and Classification

The system's requirements are categorized into **Functional Requirements (F)**, and **Quality Requirements (Q)**. These are further subdivided into relevant categories.

Functional Requirements: Functional requirements specify the system's capabilities and features, focusing on the key technologies used in this project.

Database Curation

- **F1:** Tables must have an identification tag system to facilitate similarity search.
- **F2:** Tables must include a natural language description that explains the structure of the table and its relationships within the database.

Prompt Engineering

- **F3:** The system must simulate a hypothetical response to user queries before initiating the similarity search.

Fine-Tuning

- **F4:** The system must be fine-tuned according to the specific use case, industry, or application module.
- **F5:** The system must be trained to use a JSON format when a function call is required.
- **F6:** The system must <u>not</u> be fine-tuned using proprietary or personal data.
- **F7:** The system must be retrained using correct Retrieval-Augmented Generation (RAG) responses to improve accuracy.

Agentic Behavior

- **F8:** The system must process function calls in JSON format, coming from the large language model (LLM).
- **F9:** The system must handle user authorizations and roles, integrating them with the ERP.
- **F10:** The system must determine the relevant module based on the task.
- **F11:** The system should support multi-step reasoning to execute complex tasks.
- **F12:** The system must have retrospective capabilities to check if the response needs improvement.

RAG System (Retrieval-Augmented Generation)

- **F13:** The system must incorporate an embedder model to translate natural language queries and external knowledge into vector representations.

- **F14:** A vector database must be employed to store proprietary and external knowledge (e.g., from the internet) in vectorized formats.
- **F15:** The system must execute functions within the ERP.
- **F16:** The system must support a wider context length to avoid catastrophic forgetting.

Interface Requirements

- **F17:** The system must provide an API that allows integration with various external applications.

Non-Functional Requirements Non-functional requirements are divided into categories based on user feedback and overall system comfort, ensuring user satisfaction and ease of use.

Feedback

- **Q1:** ERP stakeholders vary widely in technical knowledge, and many users are older.
- **Q2:** A feedback button should be included to report problems or submit feedback.
- **Q3:** An expandable menu should be provided, allowing users to view the system's thinking and action process (e.g., SQL commands, retrieved tables, results).
- **Q4:** A star-rating system should be implemented, allowing users to rate the system's responses.
- **Q5:** A reranker should be added to sort the responses and retrieved data by relevance.

Comfort

- **Q6:** The system should be easy to access but non-intrusive for users.
- **Q7:** It should appear as a small icon that expands only when the user interacts with it.
- **Q8:** The system should offer voice-command options for accessibility.
- **Q9:** It should support the processing of PDF and image formats.
- **Q10:** The system must be able to detect the prompt language and translate table content if the data is in a different language.

4.3 Requirements Evaluation

The requirements have shown that these systems are more capable than assumed. Complementing the model with tools allows it to overcome concerns about accuracy and privacy. The key technologies are the RAG system and function calling.

Of course, the development effort must not be understated. The number of tables in an ERP system can reach six figures, and adapting them to become AI-enabled presents the bulk of the work. The authorization concept can be transferred by adding user identification to the prompt. A JSON format could be envisioned including all relevant information.

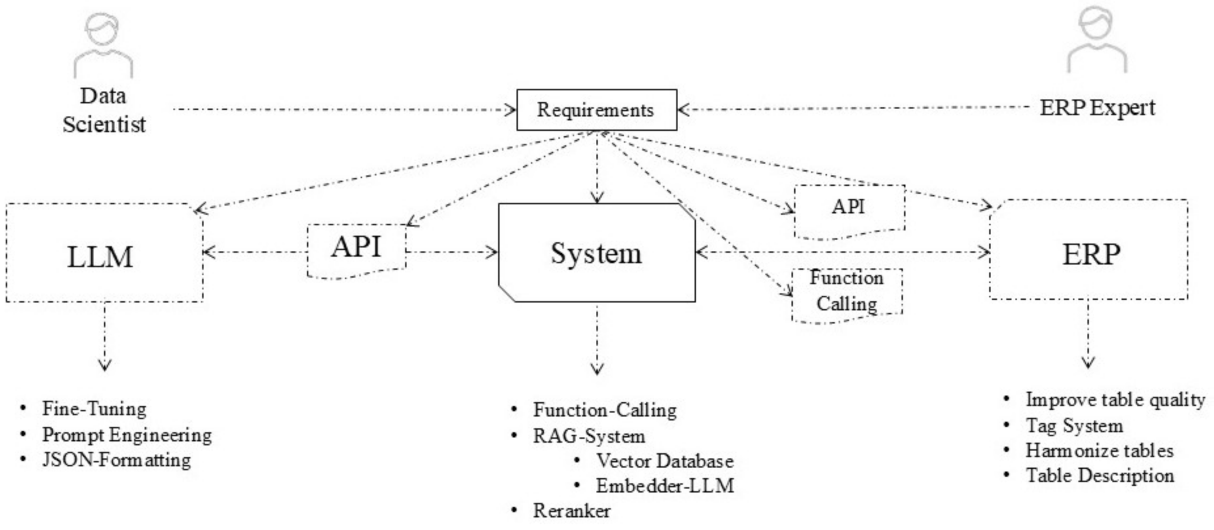


Figure 4.2: System Overview

Figure 4.2 shows the general architecture of how the three systems work together. The foundational model delivers the customized prompt via API to the RAG system. This interprets all the elements: user, desired content, and the hypothetical answer. The system then recognizes the tools to use and starts the introspective ReAct process of retrieving relevant data, using tools as it refines its final response. The user accesses the system through the ERP using the embedded API. The integration relies on customization on both ends and the complete development of the interface.

The default use case for this tool is data manipulation. Tasks like report generation, language translation, redacting of emails, and insights discovery. When it comes to task automation (e.g. automatic booking of flights, orders, etc.) it might take future model iterations with much higher accuracy.

5 Conclusion

This study has highlighted the various challenges that must be addressed to meet stakeholders' expectations within ERP environments, particularly as foundational models continue to evolve. The current state of these technologies, while promising, still faces significant hurdles that might exclude them from critical use cases or from processing certain types of data. From technical limitations to regulatory concerns, the path toward seamless adoption is far from straightforward. However, with continuous development and investment, the industry is witnessing groundbreaking innovations that promise to redefine how businesses operate.

Companies, public institutions, and universities worldwide—especially in developed nations—are dedicating immense resources to tackle the challenges inherent in these technologies. This broad, collaborative effort means that even papers like this one risk becoming outdated almost as soon as they are published. It is crucial to recognize that the landscape is evolving continuously, and new developments emerge almost daily.

On the other hand, the challenges that accompany such fast-paced growth cannot be underestimated. Whether it's technical integration issues, scalability concerns, security vulnerabilities, or the complexities of aligning with varying regulatory frameworks, there are numerous obstacles that businesses must navigate to successfully implement these technologies. For many stakeholders, the learning curve is steep, requiring both technical expertise and an understanding of the broader market dynamics. Furthermore, businesses must adapt quickly to keep up with these changes, or risk being left behind as more agile competitors embrace the future of technology.

The potential for these technologies to streamline operations, enhance user experiences, and deliver unprecedented value is immense. For SaaS providers, this presents both a challenge and an opportunity. Those who can adapt and leverage these innovations will be well-positioned to lead in the next wave of technological evolution. The competitive advantage for early adopters of cutting-edge solutions, such as AI, machine learning, and cloud-based services, cannot be overstated.

Finally, stakeholders must stay more informed and adaptable than ever, as the competitive landscape will shift dramatically. Collaboration between industry leaders, regulators, and academia will be essential to overcoming the most pressing challenges, ensuring that the benefits of these technologies are widely accessible and that they are deployed ethically and responsibly.

A Appendix - ERP Expert Interview

Interview Protocol

Project Title: Discovering General Requirements for Integrating LLM-Based Applications into ERP Environments

Goal of the Interview: Discover the general requirements ERP are subject to

Questionnaire

Introductory Questions

1. Can you please introduce yourself and describe your expertise in ERP systems and/or LLMs?
2. How familiar are you with LLMs and LLM-based tools or applications? How integrated are these into your professional life?
3. How familiar are you with integrating AI into ERP software or a business context?

ERP

4. How do users currently process and analyze tabular data in an ERP system? Are there any repetitive tasks or data interpretation issues they frequently encounter? What challenges do they normally face when it comes to large sets of tabular data?
5. When handling large or complex datasets in an ERP, are there any limitations that make decision-making slower or more difficult for users?
6. Do users face any challenges when querying insights from large datasets within the ERP system?

LLM Integration

7. Do you see any clear context where LLMs could potentially create value in an ERP environment? What about detrimental?
8. Which of this options do you think are more implementable:
   a. Chatbot that can retrieve and synthesize vast amounts of tabular data and generate reports.
   b. Agent that can automate and operate tasks and react to conditions.
   c. LLM-based software that can analyze big data pools and recognize patterns.
9. What processes within the ERP environment require users to perform manual data manipulation or reporting? Could this be improved by automating complex queries or generating insights through natural language input? Would ERP stakeholders find value in using natural language queries?
10. Can you identify specific roles where users struggle with data interpretation, and how could an LLM help them make better decisions?
11. How important is exactitude when retrieving data? Do you see clear contexts in which exactitude is more / less critical?
12. What is the rate of data generation in a business context? In other words, how much quantity of new data must an ERP be able to handle on a normal basis? How important is real-time data analysis in a business context?

Security and Compliance Considerations

13. Does ERP handle critical data (e.g. financial data, personal information) differently? How is this achieved?
14. When it comes to user access and role-based permissions in SAP, how do you ensure that only authorized users interact with critical data?

15. Do you encounter any challenges with data quality or consistency in tabular datasets within the ERP?
16. How frequently do ERP users face errors or discrepancies in reports generated from tabular data? How error-sensitive is this context

Expert Interview

Expert - Jannik Sonnenmoser
September 17, 2024, 2:05 PM - Weingarten, Germany
Duration: 39 Minutes 20 Seconds

Interviewer: *Can you please introduce yourself and your expertise in ERP systems and/or LLMs?*

ERP Expert: *Yes, my name is Jannik Sonnenmoser, I am 25 years old, and my experience with ERP systems is primarily with SAP. During my studies, I worked for three years in the technical part of SAP, focusing on SAP Basis, which involves the technical background. For the past two years, I have been working as a software developer within the SAP environment, mainly in cloud solutions but also in classic ABAP development. My expertise with LLMs (Large Language Models) is limited to personal use. I use them myself and am interested in how SAP plans to incorporate these technologies in the coming months and years.*

Interviewer: *You mentioned you use LLMs or AI in your personal life, but not so much in your professional life. Is that correct?*

ERP Expert: *That's not entirely accurate. I also use LLMs professionally, especially for code generation while programming. However, the SAP-specific knowledge in models like ChatGPT is still quite limited, so I use LLMs for various tasks where their current SAP knowledge is useful.*

Interviewer: *So you are not familiar with any specific implementations or business contexts within SAP that use LLMs?*

ERP Expert: *I am aware of some implementations, but I do not work with them directly. There are various SAP products that use AI, such as SAP Ariba for procurement, which uses AI to predict future needs. However, I do not use these specific applications. The only AI-related tool I might interact with is SAP's own AI initiatives, but I have not yet encountered them in our systems.*

Interviewer: *Moving on to the ERP part, could you describe how users currently process or analyze tabular data in general within SAP?*

ERP Expert: *In SAP, users often handle large amounts of data. For instance, material master data or customer orders can involve tens of thousands of records. This is typically managed through analytical apps or dashboards that provide an overview of the data. These tools consolidate and display large data sets effectively.*

Interviewer: *Are there repetitive tasks or challenges that users face in this context?*

ERP Expert: *Yes, there are many repetitive processes in SAP. For example, tasks related to handling routine processes. While process optimization aims to reduce repetitive tasks, it is still a significant part of the workflow. Performance issues* *and data quality problems are common challenges. If the data is incorrect or incomplete, it complicates the analysis and reporting processes.*

Interviewer: *Do you see LLMs or AI as helpful in this context, perhaps in automating repetitive tasks?*

ERP Expert: *AI can be particularly helpful in guiding users through tasks they perform infrequently. For example, if someone is entering a customer order for the first time, AI could assist by providing guidance or correcting errors.*

Interviewer: *Can you provide examples where LLMs or AI could be beneficial in the SAP context?*

ERP Expert: *AI could significantly improve efficiency in areas like procurement by predicting requirements based on market trends. Additionally, AI could assist in software development by reducing the time and cost involved. For instance, SAP Build, a low-code/no-code platform, could be enhanced by AI to allow users to create apps by simply describing their requirements, even without technical knowledge.*

Interviewer: *What are some challenges or limitations you see in integrating LLMs with SAP systems?*

ERP Expert: *One challenge is the integration of AI into SAP systems where each client may have custom fields or unique requirements, which complicates the development of analytical apps. Additionally, while standard SAP functionality can handle many tasks well, customization and integration with AI can increase complexity and development time.*

Interviewer: *Do you think there is potential for further advancement in using LLMs or AI within SAP?*

ERP Expert: *Yes, there is significant potential. AI could enhance various aspects of SAP systems, from improving data handling to automating repetitive tasks. For example, AI could streamline the procurement process or support developers by making app creation more intuitive and less dependent on technical expertise. Although SAP has made strides in this area, there is still room for growth and innovation.*

Interviewer: *Are there any specific areas where you see potential for AI, especially LLMs, to help with system integration and user interactions in SAP?*

ERP Expert: *One possibility is using AI to help users who might have difficulty using traditional input methods. For instance, instead of using a keyboard, users might interact with the system through voice commands or image recognition. This could be beneficial in scenarios where users have physical limitations or are in environments where traditional input methods are impractical. For example, in a warehouse setting, if AI could recognize barcodes through cameras and automatically process them, it would save significant time compared to manual scanning.*

Interviewer: *Do you have any practical examples of how AI has improved processes in a real-world scenario?*

ERP Expert: *Yes, during my time working at Ravensburger, I encountered a practical example. In their warehouse, each package had to be individually scanned for shipment, which was time-consuming. Implementing a system with cameras that could read barcodes and process the data automatically significantly reduced manual effort and time. This is an example of how AI can improve efficiency and accuracy in repetitive tasks.*

Interviewer: *What role does data quality play in the context of using AI with SAP systems?*

ERP Expert: *Data quality is crucial. If AI systems rely on inaccurate or incomplete data, it can lead to significant issues. For instance, if an AI system mistakenly orders 1001 items instead of 1000 due to data errors, it could cause problems. Ensuring high data quality and having checks in place to validate AI outputs are essential. Even though AI can assist in data management, human oversight is still necessary to ensure accuracy.*

Interviewer: *How do you see AI assisting in data handling and reporting?*

ERP Expert: *AI can be useful in improving data handling by identifying and correcting errors or gaps in the data. For example, AI can help with data entry by prompting users to complete mandatory fields or alerting them to potential errors. However, post-entry data corrections might still require manual intervention, especially if data needs to be validated against additional sources.*

Interviewer: *Can you discuss the importance of user acceptance and how it affects the implementation of AI tools?*

ERP Expert: *User acceptance is critical. AI tools must be user-friendly and align with the needs of diverse users, from tech-savvy millennials to those with decades of experience. For instance, a chatbot or AI tool needs to be intuitive enough that even users who are not comfortable with technology can adopt it. Ensuring that the AI tool is accessible and provides clear benefits is key to its successful implementation.*

Interviewer: *How do you handle data security and access control in relation to AI and SAP systems?*

ERP Expert: *Data security is a major concern. AI tools, such as chatbots, should not have access to sensitive personal data or financial information. SAP systems use a role-based authorization model to ensure that users only have access to the data they are permitted to see. Implementing strict access controls and ensuring that AI tools comply with these controls is essential for maintaining data security.*

Interviewer: *How sensitive are users to errors in reports and data presented by SAP systems?*

ERP Expert: *Users are highly sensitive to errors, especially when the data is critical for business decisions. If a report contains incorrect data, it can lead to incorrect conclusions and potentially costly mistakes. For reports used by management or other critical business areas, accuracy is paramount. Users expect reports to provide reliable and actionable insights, and any significant errors can undermine trust in the system.*

Interviewer: *In terms of AI and SAP, what are some key considerations for improving data quality and user experience?*

ERP Expert: *Key considerations include ensuring that AI tools are easy to use and understand, providing clear benefits to the users, and integrating well with existing SAP systems. Improving data quality involves not only using AI to catch and correct errors but also ensuring that the input data is accurate and complete. User experience can be enhanced by making AI tools intuitive and providing adequate training and support to users.*

Interviewer: *Thank you for your time*.

B Appendix - Data Scientist Interview

Interview Protocol

Project Title: Discovering General Requirements for Integrating LLM-Based Applications into ERP Environments

Goal of the Interview: Discover the general requirements for a LLM-Based Application to Successfully Process Tabular Data

Questionnaire

Introductory Questions

1. Can you please introduce yourself and describe your expertise in ERP systems and/or LLMs?

   My name is Nils Krause, currently working as an intelligence data engineer at Airbus with 5 years of working experience in data science. Last year I created in that working environment a RAG system enabling precise context searches in huge text files. There I gathered extensive expertise in LLMs and the technical knowledge to create and deploy such systems at scale. My knowledge concerning ERP systems covers the fundamentals.

2. How familiar are you with integrating AI into ERP software or a business context?

   Developing and Integrating AI systems for enabling business in Airbus has been my main task there. Integrating AI in ERP systems I haven't had any experience yet but the AI toolset integrating AI in ERPs remains mostly the same like in others business contexts.

Large Language Models

3. What do you see as the primary strengths and limitations of current LLMs in handling structured data?

   The primary strength of LLMs in handling structured data is that their underlying architecture of LLMs still follows the scaling laws and thereby will continue to improve with model sizes and amount of data injected. Recent developments in function-calling and JSON formatting integrated for example in OpenAI LLM systems is demonstrating that these systems are quite well at handling structured data for in and output. Still, the current state-of-the-art systems do not reach 100% accuracy so there is still room for improvement.

4. What are the main challenges that arise when dealing with tabular data?

   One of the main challenges in handling tabular data is the amount of tabular data that can be recalled directly from the LLM due to context size and the tendency to lose more information with growing context lengths. Google's new Gemini model with 1M tokens keeps pushing on that context lengths problem.

5. Based on your experience, what are the common pitfalls or challenges one might face when integrating LLMs with structured data?

One of the many pitfalls in integrating LLMs on structured data is the underlying data format of your database, that might be different from one table to another leading to suboptimal performance when deploying one model on a different range of data formats. Instead of building one model for all your structured data you can go with multi modal models for each specific data format.

Tools for Leveraging LLM Capabilities

6. What tools or frameworks have you used or are aware of that facilitate the integration of LLMs with structured data?

   The CRISP-DM framework can help you structure your project but specifically for LLMs.
   Additional Tool Frameworks: Langchang, reasoning, Open Source Reranker and LLMs or for simplicity OpenAI API

7. In your opinion, what features or capabilities should these tools possess to be effective?

   An underlying understanding of the domain they are operating in.

Requirements Engineering

8. Can you provide examples of successful requirements engineering processes you've been involved in for similar projects?

   No I cannot.

9. What key functional requirements need to be addressed to enable an LLM-based solution to work with tabular data?

   Search for relevant tables (Vector DB, Reranker), Create a query, Execute Query and Function Calling.

10. What non-functional requirements (e.g., accuracy, security, compliance) should be considered for such an application?

    Data Compliance when using Hosted LLMs, error handling, triggering new try cycle.

11. How would you prioritize these requirements, and what factors influence this prioritization?

    It is a pipeline, input and output, special focus on orchestrating these models and evaluating the subcomponents.

12. Are there any industry-specific standards or best practices that should be considered when defining these requirements?

    Not that I know of.

13. Can you provide examples of successful integrations of AI/LLM solutions in ERP systems that you are aware of?

    Not Currently.

Future Trends

14. What future trends or developments do you foresee in the field of LLMs and their ability to handle structured data?

    Stronger data pipelines like langchain that give the model different tasks to look at a problem combined with reasoning capabilities (OpenAI ChatGPT-o1).

15. How do you envision the role of LLMs evolving in the context of enterprise data management?

    To better understand what the model is aka. Transformers and Andrej Kapathy.

16. Would you consider reviewing my paper before publication?

    Yes for sure, at least and quick review.

<u>Expert Interview</u>

Expert - Nils Krause
September 25, 2024, 5:53 PM - Ravensburg, Germany
Duration: 45 Minutes 50 Seconds

Interviewer: So let's maybe jump directly into the general thing I'm thinking of. From my perception of your project, it seems like you have an SAP system where you have data stored, such as product information, sales data, warehouse information, and so on. You want someone working somewhere in the company to know how well the company is performing, so you need different tables, like delivery data, manufacturing data, and so on. Is that correct?

Data Scientist: Yes, exactly. So you have this huge SAP database with multiple tables. I was thinking of a system where you first ask a natural language question, for example, "I want to get all the revenue for this month or for this quarter."

Interviewer: Right.

Data Scientist: Then the next part would be search. So you ask the question, "I want to get the revenue for a quarter," and now the system needs to search for the related tables because there are thousands of tables in your database. First, it involves a search, and from my experience with building retrieval-augmented generation systems, it works on text comparison.

Interviewer: Interesting.

Data Scientist: Yeah, you calculate the similarity between sentences or between a sentence and a chunk of text. The system retrieves the most relevant results. There's a technical step that improves it further — instead of just comparing sentences directly, you can translate a sentence into a numerical representation, like a vector. This numerical representation can then be compared to the vectors of larger text chunks in the database.

Interviewer: Yeah.

Data Scientist: So the system compares these two vectors and finds the most similar text chunks related to the question.

Interviewer: Yeah, that’s exactly the process. I explained something similar in my theoretical section, and I mention that retrieval-augmented generation (RAG) is one of the most important tools.

Data Scientist: Yes.

Interviewer: And it's great because you have a much more practical approach than I do. I’m still learning all these concepts, and the way you explain it makes it easier for me to understand.

Data Scientist: Hmm.

Interviewer: You clearly have a much deeper understanding of these concepts, and it's really helpful to see your perspective.

Data Scientist: But I think you will also gain this insight when you build a system on your own. I built one system, and when you're building something, you develop a different connection to the theoretical part. While coding and trying to execute queries against the database you're working with, you start to understand things on a different level. But starting with a high-level understanding, like "What kind of steps do I need to take to build the system I want?" is crucial. Then you move into the technical approach, like deploying your vector database, inserting information into it, querying it, and so on. You take it step by step.

Interviewer: That’s great. So, basically, what you just explained is the use case. What would be the use case for the LLM?

Data Scientist: Yes, but I haven’t finished yet. So, the first part is getting the relevant tables.

Interviewer: Right.

Data Scientist: And that’s the first challenge, because you often only have the table names, and there might not be any descriptions of what these tables contain. So, the first problem with the search is that you need to enrich the tables with information, like descriptions. Without these, it's hard to compare the text. In an SAP system, for example, there might be complicated, non-descriptive terms. The first step would be to enrich the tables with descriptions or keywords, so you have text to compare.

Interviewer: So, you mean the challenge is more about context optimization? The tables may need to be adapted so that the LLM can better understand or perform the search?

Data Scientist: Yes, exactly. In the first part, it's all about search. To get good search results, you need a list of tables with some textual information attached—maybe tags or descriptions—so that you can search effectively.

Interviewer: Got it.

Data Scientist: Once you've found the correct table, the next step is understanding the table structure—rows and columns. For example, if you want to calculate the sum of revenues for a quarter, you need to perform an SQL query. After finding the table, you then need to determine the operation to execute based on the input question. This is quite a complex problem.

Interviewer: That’s OK. The more complexity you explain, the more requirements I can extract.

Data Scientist: Thank you. I’m glad it's helpful.

Interviewer: It is! You're mentioning things I hadn't even thought of yet, so it's really valuable.

Data Scientist: OK, so from the question, you not only need to find the table, but you also need to determine the user's intent. For instance, does the user want to retrieve the revenue column for the last three months or the last quarter? Then you have to translate this intent into an SQL query. After formulating the query, you need to ensure that the SQL statement is correctly formatted, especially if this process is automated. You don’t want errors, like incorrect brackets, to break the query. In OpenAI's API, they call this "function calling" using JSON format. To perform function calling, you need a correct and consistent schema.

Interviewer: Right.

Data Scientist: So, when you execute the SQL query, it doesn’t return an error.

Interviewer: Yes.

Data Scientist: Once the query runs successfully and returns a result, like "The revenue for the last quarter is 1.2 million," you need to feed this information back into the system. A large language model can then take the result, the input question, and maybe some additional context, and generate a solid answer for the user.

Interviewer: I see. First, you get the revenue, then there are other steps... I can’t remember all of them, but you’re explaining them well.

Data Scientist: Exactly. You have the search, which is the first part. Then you have to understand the user's intent.

Interviewer: Yes, exactly.

Data Scientist: And from the user's intent, you have to extract the SQL statement you want to execute. While executing the query, you need to ensure it has the correct format. Once the SQL query is executed, you get the answer back. Then you combine the answer with the context and the question, put it into a text block, and print out a response for the user.

Interviewer: So the process would be: search for the correct tables...

Data Scientist: Yes.

Interviewer: ...and enrich the tables with tags or descriptions, right?

Data Scientist: Yes.

Interviewer: Then recognize the operation the user wants to perform, which would require a tool or function call in JSON, like you mentioned for OpenAI. Afterward, it performs the operation or retrieval, and then this has to be formatted into...

Data Scientist: Mm-hmm.

Interviewer: ...the response, and delivered to the user.

Data Scientist: Yes.

Interviewer: OK.

Data Scientist: That's the general system I'm envisioning for how you can build this kind of system.

Interviewer: I see. For example, I'm looking at OpenAI's blog post or documentation called *Optimizing LLMs for Accuracy*. They have an LLM optimization matrix, and at the bottom, there's prompt engineering, fine-tuning, RAG, and a combination of some of them.

Data Scientist: Mm-hmm. Right.

Interviewer: So, I'm following this matrix, more or less. To put what you're saying into this context, could you talk a bit about how much prompt engineering, fine-tuning, or RAG would be required?

Data Scientist: I think the main part is RAG, and to improve RAG, prompt engineering is essential. For instance, when you have a question and you're looking for the corresponding text in a database, the challenge is that questions are often semantically different from answers.

Interviewer: Yeah.

Data Scientist: One way to find better results is to take the question and have the model generate hypothetical answers. Instead of searching for the question in your text data, you search for the hypothetical answer that the language model produced. This is because...

Interviewer: Go on.

Data Scientist: It's not just a belief but more of a proven method. Searching for hypothetical answers often yields better results than searching for the question itself.

Interviewer: So, the human generates a prompt or question, and before running that through the whole RAG process, prompt engineering reformulates the question into something that makes more sense or is structured better?

Data Scientist: No, not quite. There's still the original question, but the question itself isn't similar to an answer. For example, if someone asks, "How old is my dog?"...

Interviewer: Yeah.

Data Scientist: There's something like "My dog is 18 years old" or "My dog is 10 years old." The semantics, like the order of the words, are different. So, "My dog is 15 years old" and the question "How old is my dog?" are different. When you're directly looking for the question in your database, you might not get the most relevant answer. But if you give the question "How old is my dog?" to an LLM and say, "Please provide a hypothetical answer to my question,"...

Interviewer: OK.

Data Scientist: ...and the hypothetical answer is "My dog is [X] years old," you then take that answer...

Interviewer: Yeah, and then use that response for the search.

Data Scientist: Exactly. This is a way to use prompt engineering to improve the retrieval-augmented generation (RAG) system.

Interviewer: That's pretty clever.

Data Scientist: So that would be one fine-tuning step. For example, once you get the table you need, you find the user's intention, then you want to formulate it into an SQL query. If you try to execute the SQL query and it returns an error, you can take that error and feed it back to the LLM. You can say, "I wanted to do this, but I got this error, please fix it and rewrite the query."

Interviewer: So, it's almost like the system acts as an agent that can reason and introspect—kind of like talking to itself. Is that correct?

Data Scientist: I haven't looked into the exact definition of an agent, but I'm aware of a library called LangChain. LangChain orchestrates different text flows.

Interviewer: Yeah, yeah, exactly.

Data Scientist: So, in theory, you have RAG, and in practice, you have tools like LangChain. It's looking at the same problem from two sides: theoretical and practical.

Interviewer: Yeah, OK, I get it. So, you use agents to improve the process. And then, with prompt engineering, the better the prompts, the more the RAG system improves overall.

Data Scientist: Yes, exactly.

Interviewer: So, the system would formulate an SQL command, and in case there's a mistake, it wouldn't deliver the error to the user right away. It would first go through a feedback loop, trying to fix the problem by itself.

Data Scientist: Yes, exactly.

Interviewer: And if it still doesn't work, it would prompt the user with a natural language response, like, "Can we try something else? Can you reformulate the question?"

Data Scientist: Yeah, exactly.

Interviewer: Just one more thing about fine-tuning. I had an interview with a friend who works for my company, and he's an expert in ERP systems. He doesn't know much about AI, but he mentioned that in areas like production, new data is created every day—thousands of new entries. But not every use case needs that constant data influx. However, ERP databases are generally constantly updated. So, does that mean we always need to fine-tune the model?

Data Scientist: When it comes to fine-tuning, you have to think of it as... You have parametric knowledge, which is the knowledge embedded in the LLM itself. When you do a zero-shot prompt—where you simply ask a question, like, "How do I..."

Interviewer: Yeah.

Interviewer: Get the first 10 numbers of π or so, yeah. Then this model has intrinsic knowledge about the world and gives information. When you want to enrich that information, the parametric knowledge of your model, then you fine-tune it.

Data Scientist: Yeah.

Interviewer: Then fine-tuning is a good idea. For example, when you want to put more information about some concepts. Like, for example, when you're working in a very specific area of medicine where there are specific definitions and concepts, like the brain or something else that the model doesn't know. When you want to find the intention behind the user's query by asking the large language model, and the model doesn't understand the context, it won't be able to provide relevant information. But fine-tuning at the beginning stages, like when you're starting with a retrieval-based system, doesn't play a huge role. It's more important at the very end, for example, with ERP systems, where the content is more standard and often already covered by the base models. Fine-tuning at the start might not matter much.

Data Scientist: Yeah.

Interviewer: OK. But for specific areas, maybe like therapy software used by every industry, or for small changes like laws in one country, or in biochemistry or even the military, where they might use an intranet, I guess there’s some room for that?

Data Scientist: Hmm. Bye.

Interviewer: But also, there are many security and compliance implications to think about.

Data Scientist: Yes, and for example, when you have knowledge that's constantly changing, like where people are located. If you're currently seeing an address book, that can change over time. And in cases like this, it's tricky.

Interviewer: You can't fine-tune parameters every time someone takes a trip.

Data Scientist: Exactly, it's very expensive. You should focus on improving the parametric knowledge for more stable information. For changing contexts, integrate it as non-parametric knowledge, for example, through a retrieval-based argument generation system.

Interviewer: Yeah, I had one more question. What about numerical data? Some contexts need accuracy, especially with lots of numbers in tables. Can you rely on LLMs for that?

Data Scientist: That’s a challenge for large language models. In earlier models, if you asked them to count words in a sentence, they performed poorly because they're trained to predict the next token, whether it's a word, character, or number. For handling numerical tasks, I recommend letting the LLM write the mathematical code, like in Python, and then execute it rather than relying on the model directly.

Interviewer: So you’d recommend using a function call as soon as possible and passing it to a specialized tool instead of letting the LLM handle it by itself?

Data Scientist: Yes, exactly. For basic reasoning, like simple addition, it might work, but for complex tasks like finding averages or performing more intricate operations, it’s much better to use an external component to execute the code.

Interviewer: OK. I also wanted to ask, what do you think about compliance and security, particularly authorizations? Any thoughts on the requirements?

Data Scientist: Hmm.

Interviewer: For instance, if I'm a regular user of a system like therapy and I ask for private information about my postal data, a CEO shouldn't have access to the entire database.

Data Scientist: Yeah, that's a major concern. You need identity and access management to control who has access to what. When you deploy this for enterprise solutions, you must define user roles, so they only have access to specific columns and tables based on their permissions.

Interviewer: So that involves some form of prompt engineering? For example, if I'm logged in with my username and password, the system knows my role and only returns information that matches my authorization, right?

Data Scientist: Yes, exactly. Identity and access management won't be handled by the LLM but by an external system. When you log in, the system knows your role and limits your access to the database accordingly.

Interviewer: So the database in this case is a vector database, while the LLM's parameters only cover general or specialized knowledge like grammar or word associations. It wouldn't be trained on private corporate data.

Data Scientist: Right.

Interviewer: The parameters wouldn't be fine-tuned for private data but rather for understanding general concepts, and the retrieval system would handle the specifics. Did I get that right?

Data Scientist: Yes, exactly.

Interviewer: And my access would be limited to my role and authorization.

Data Scientist: Yes.

Interviewer: That's really useful information. Thank you so much!

Interviewer: So that's solved. But what about hardware or costs?

Data Scientist: The cost of running such a system?

Interviewer: Yeah. Could you talk a little bit about that?

Data Scientist: Sure. I hosted a vector database called Qdrant, which I tested with a limited amount of data. They offer a free tier, so if you're just starting out and experimenting with a small amount of data, it's inexpensive. Even if you're using a cloud provider like Google Cloud Platform (GCP) or AWS, hosting the database won't cost much initially.

Interviewer: But let's say a large company like Airbus with 70,000 employees wants to implement a chatbot for them to make inquiries. Wouldn't that require a lot of resources?

Data Scientist: Yes, for a company of that size, it will become expensive, especially if you're hosting the system on your own infrastructure. Maintaining your own cluster could be costly, especially in the short term. On the other hand, using a cloud provider might be cheaper initially, but as the number of users grows, so does the cost. If you have 100,000 employees making several requests daily, those expenses can add up.

Interviewer: Yeah.

Data Scientist: But at the same time, more usage means the system is providing value. The more requests, the more efficient the employees can become. Instead of navigating complex systems like SAP, they could get direct access to the information they need. While I can't give you exact costs, it's worth noting that starting small with tools like OpenAI's API and Qdrant can be done at almost no cost.

Interviewer: That's good to know. So for development or optimization, the cost is close to zero.

Data Scientist: Yes. When you're just starting to experiment with technical implementation, you can use free trials, OpenAI's API, and other tools to get a feel for how the system works. Once you see real benefits, that's when corporate and financial stakeholders would get involved to make larger decisions. But that comes later, after the prototype phase.

Interviewer: Yeah, so your focus is on the experimental phase.

Data Scientist: Yes, exactly. I'm more involved in the experimental side of things.

Interviewer: I have just two more questions. One is about quality requirements. Is there anything that could make the tool easier to use or add value in terms of performance?

Data Scientist: Do you mean in terms of the user interface or performance improvements?

Interviewer: Either, or both.

Data Scientist: Well, one thing would be adding a reranker. After searching the vector database, a reranker could look at the top results and sort them by relevance. This would improve the search results. Additionally, using an agent with something like LangChain could enhance the quality, though it might increase execution time as it goes through more cycles of reasoning.

Interviewer: Yeah.

Data Scientist: From an interface perspective, allowing users to upvote or downvote the answers they receive could also be helpful. This would give feedback to improve the system by tracking performance in a production environment.

Interviewer: Maybe.

Interviewer: Then having the ability to track which user queries led to satisfied users and which led to unsatisfied ones would be very beneficial for future improvements.

Data Scientist: Yes, knowing which parts of the system need improvement would be very useful.

Interviewer: Maybe, like Perplexity AI does with sources, there could be an option to show which tables were involved in generating the answer. You could provide a normal view, but if the user clicks a button, it could display the SQL queries or the involved tables.

Data Scientist: Yeah, that's a good idea. It would give users more control and transparency. They could check the tables themselves or modify filters if they think something went wrong. This gives more flexibility.

Interviewer: One last question. I saw that SAP is launching its own LLM-based chatbot, and they claim it can handle tasks like booking flights just by telling the chatbot, "Book me a flight for this date." Is that too complex for a chatbot?

Data Scientist: No, it's quite possible. It’s similar to function calling. The model would understand your intent and instead of just running a database query, it could also handle payment processes like taking your credit card details and completing a booking. It’s not too far from what’s already happening.

Interviewer: How comfortable would you personally feel trusting an LLM to book a flight for you?

Data Scientist: Honestly, not very comfortable. I’d worry about my credit card info and would want to double-check train connections, prices, and other details myself. While LLMs can automate this, many people still prefer a personalized experience, like going to a travel agency for an all-inclusive package. For more individualistic travelers, it might still require manual research and comparison.

Interviewer: OK, I see. I can use that for my project.

Data Scientist: Great! I also noted down some key points for your questionnaire. We didn’t go through them all one by one, but here’s a summary with the fundamental answers to the topics you mentioned.

Interviewer: Thanks a lot!

References

Thimira Amaratunga. *Understanding Large Language Models: Learning Their Underlying Concepts and Technologies.* Apress and Imprint Apress, Berkeley, CA, 1st ed. 2023 edition, 2023. ISBN 979-8-8688-0017-7. doi: 10.1007/979-8-8688-0017-7.

Kenneth Ward Church, Zeyu Chen, and Yanjun Ma. Emerging trends: A gentle introduction to fine-tuning. *Natural Language Engineering*, 27(6):763–778, 2021. ISSN 1351-3249. doi: 10.1017/S1351324921000322.

Stefan Feuerriegel, Jochen Hartmann, Christian Janiesch, and Patrick Zschech. Generative ai. *Business & Information Systems Engineering*, 66(1):111–126, 2024. ISSN 2363-7005. doi: 10.1007/s12599-023-00834-7.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. URL `http://arxiv.org/pdf/2312.10997v5`.

Andrea Herrmann. *Grundlagen der Anforderungsanalyse: Standardkonformes Requirements Engineering.* Lehrbuch. Springer Vieweg, Wiesbaden and Heidelberg, 2022. ISBN 978-3-658-35460-2.

IREB. *Handbook for the CPRE Foundation Level according to the IREB Standard*, 1.2.0 edition, 2024. URL `https://www.ireb.org/en/downloads/#cpre-foundation-level-handbook`.

Noriaki Kano, Nobuhiko Seraku, Fumio Takahashi, and Shin-ichi Tsuji. Attractive quality and must-be quality. *Journal of The Japanese Society for Quality Control*, 14(2):147–156, 1984. doi: 10.20684/quality.14.2{\textunderscore}147.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. URL `http://arxiv.org/pdf/2205.11916v4`.

OpenAI. Optimizing llms for accuracy, 2023. URL `https://platform.openai.com/docs/guides/optimizing-llm-accuracy/optimizing-llms-for-accuracy`.

OpenAI. Hello gpt-4o, 2024. URL `https://openai.com/index/hello-gpt-4o/`.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. URL `http://arxiv.org/pdf/2304.03442v2`.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. Chatdev: Communicative agents for software development. URL `http://arxiv.org/pdf/2307.07924v5`.

Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications. URL `http://arxiv.org/pdf/2402.07927v1`.

SAP. Joule capabilities, 2024. URL `https://help.sap.com/docs/joule/capabilities-guide/features-of-joule`.

Murray Shanahan. Talking about large language models. URL `http://arxiv.org/pdf/2212.03551v5`.

Hansruedi Tremp. *Agile objektorientierte Anforderungsanalyse: Planen – Ermitteln – Analysieren – Modellieren – Dokumentieren – Prüfen*. Erfolgreich studieren. Springer Vieweg, Wiesbaden and Heidelberg, 2022. ISBN 978-3-658-37194-4. URL `http://www.springer.com/`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. URL `http://arxiv.org/pdf/1706.03762v7`.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Du Nan, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. URL `http://arxiv.org/pdf/2210.03629v3`.

Declaration of Authenticity

I hereby declare that I have independently written my project report with the topic

Discovering General Requirements for Integrating LLM-Based Applications into ERP Environments

and that I have not used any sources or aids other than those indicated.

Weingarten, 30-09-2024

Place, Date

[illegible]

Signature