Building LLM powered Knowledge-based Systems - from RAG to Multi-agent AI Platforms

Building LLM powered Knowledge-based Systems - from RAG to Multi-agent AI Platforms - Part 1

Large Language Models (LLM) have revolutionized the way we work with unstructured data. With their capacity to understand and generate text at super human level, they enable a vast number of new use cases and make it finally possible to solve problems that have kept ideas back for decades.

One of such an idea are knowledge-based systems that where imagined in the 1980s but never came to full flourish.

In this article I will summarize two years of experience in building LLM powered knowledge-based systems varying from simple Question Answering Systems, implemented as RAG workflows, to AI Agent platforms that enable the automation of complex multi-step business processes.

Ending with a short look at the current and future applications of AI Agents.

In order to pay justice to the complexity of each type of knowledge based system, this article is split into 3 parts that are published in 3 blog posts.

The first part will describe **Question / Answering Systems**, the second **Knowledge Management Systems** and the last part will explain **AI Agent Platforms**.

Each part will focus discuss the purpose of the system, how it differs from the previous one and challenges and recommendations on building and deploying it.

But first, lets start to explain Knowledge-based systems.

Knowledge-based systems

Knowledge-based Systems (KBS) have been developed as a sub field of Artificial Intelligence since the early 1980. The main components of an KBS is a knowledge base that contains information and facts and a reasoning engine that is able to derive new knowledge and perform cognitive tasks, like problem solving, based on information contained in the knowledge base.

KBS where originally build to support problem solving in very narrow domains like medical diagnoses or protein structure analysis, through the use of rules managed by experts. In the 1990s their application was extended to perform meta-learning and to incorporate unstructured data to enable automatic reasoning.

I argue, that before the rise of LLMs, we did not have the capacity to work with unstructured data and to perform automatic reasoning in a domain agnostic way.

LLMs are the key invention that therefor enables the development of KBS systems that can understand and reason over a vast corpus of knowledge. In form of AI Agents, they can be used to solve pre-define business tasks to accelerate knowledge workers.

In this article, we will look at three types of KBS systems in increasing complexity and capabilities

- **Type I - Question Answering Systems**: Support Information retrieval for an individual or a small team on a single corpus of knowledge.
- **Type II - Enterprise Knowledge Systems**: Counter the fragmentation of information in an organization, by enabling access to all knowledge within an organization from a single system.
- **Type III - AI Agent platforms**: Accelerate the work of knowledge workers by automating individual business processes.

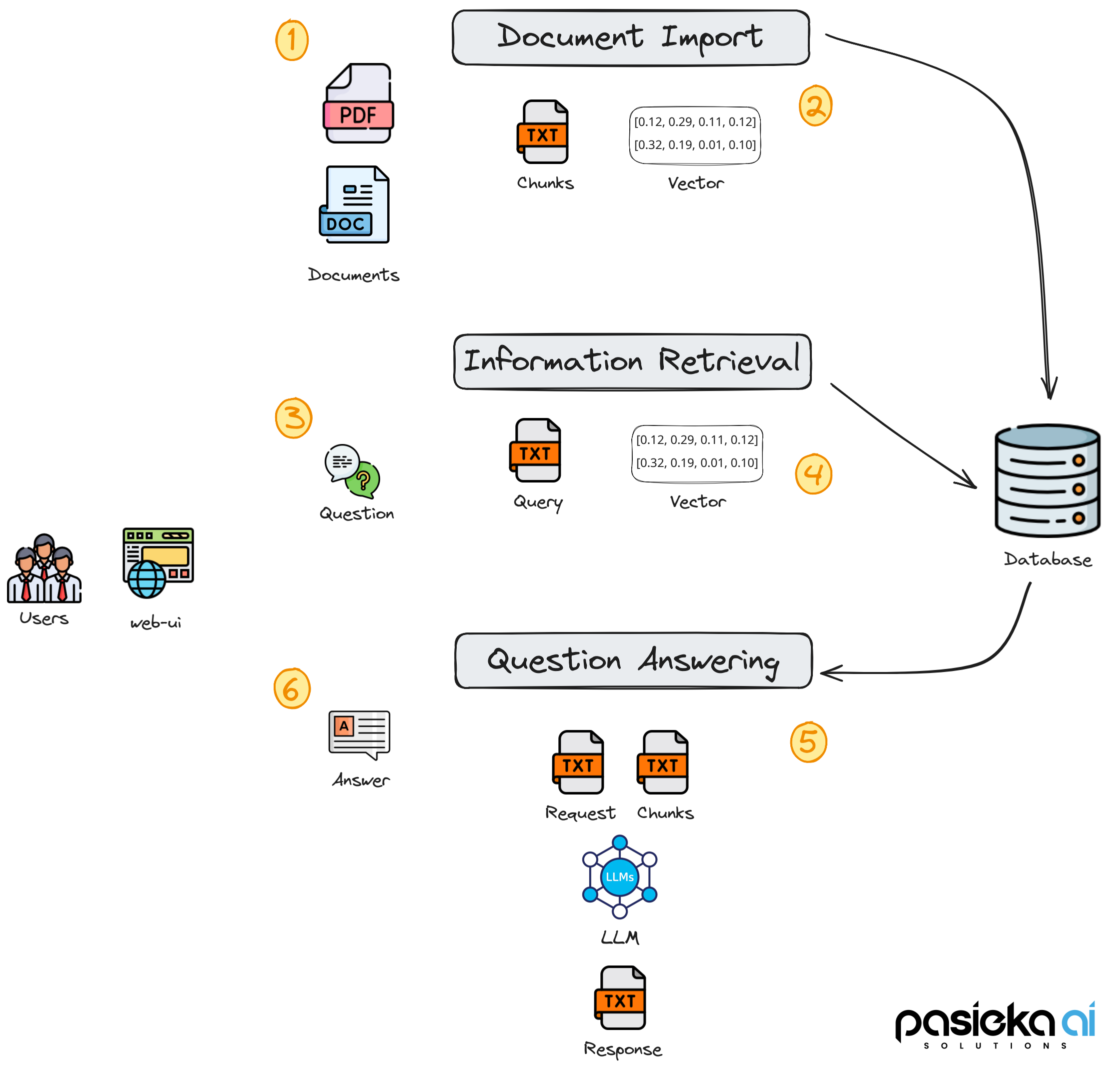
Next, we look at each of them in detail, discussing their capabilities and challenges when implementing and maintaining the such systems.

Type I - Question Answering Systems

The purpose of Question Answering Systems (QA) is to provide a small group of people a way to easily ask questions about a limited corpus of knowledge through natural language.

Although question answering systems have a vast history and the earliest systems, like Baseball date back to 1961, in this article I want to focus on Retriever Augmented Generation (RAG) systems that got popular after 2020.

Figure 1 illustrates such a RAG system that is able to Import Documents in varying formats and types (1) and performs pre-processing and storage in a database (2). Users are able to ask questions in natural language through a chat interface (3). The information retrieval component of the system identifies for a question all relevant text fragments (4) and leverages a LLM to generate responses (6) that are returned to the user in form of answers in natural language (6).



Challenges building and maintaining an Q/A System

RAG Systems have come a long way since 2020 and are today the de-facto standard way how to build Question Answering Systems.

A lot has been written on how to build RAG systems (have a look here and here) and there are many facets to building a high performing RAG system, from the data pre-processing that focuses on information representation and semantic chunking (see this reference), to hybrid search that combines keyword search and dense retrieval, re-ranking and other techniques (see this post), to select only relevant document fragments.

This being said, building your own RAG has been made significantly easier through excellent open source libraries like LLamdaIndex or LangChain that make it possible to deploy a RAG system in a few lines of python code.

Recommendations for Implementing an Q/A System

If your use case is limited to providing a Q/A system to a single team, based on a limited number of documents in standard formats (like pdf, Word, HTML, ...) there are even open source solutions that work out of the box and that can be deployed as simple docker containers, like Ragflow or Onyx (former Danswer).

Such a Q/A system can be deployed easily, making use of either local inference servers like Ollama in case of security concerns and offline use cases or cloud deployed LLMs in your own hyperscaler subscription (i.e. AWS, Azure, GCP) or LLM provider service (OpenAI, Anthropic or Aleph Alpha).

Where Q/A Systems fail

The Limitations of Q/A systems is its scale with respect to the size of the organization and the knowledge corpus that they can work with.
In addition be aware that most Q/A systems, poorly support data access policies. In most cases, documents are copied and imported to the system, providing access to all data for all users.

If your use case involves a complete organization and has to take into account (as it should) aspects of data security to incorporate varying data sources like Sharepoint, Confluence, Network shares or Email, then I recommend to have a look at Type II - Enterprise Knowledge Systems.

Key Takeaways for Question / Answering systems

- Deploy existing solutions instead of building your own.
- Be aware of the limitations of a deployed solution with respect to, data access policies, number of documents, knowledge domains and the number of users.
- Don't try to import complete document platforms, use an Enterprise Knowledge System instead.

Thank you for reading, I hope you found it useful. In the next post we will look at Enterprise Knowledge Systems that gives an organization a chance to counter the fragmentation of information by building a portal through which users can access all knowledge anywhere in the organization.

Building LLM powered Knowledge-based Systems - from RAG to Multi-agent AI Platforms - Part 2

Hello and welcome back to this three part Series on Knowledge-Based Systems. In the first part, we introduced Knowledge-based Systems and described Question Answering (Q/A) Systems that can help an individual or a group to access information about a limited domain.

Although this use-case is served well with Q/A Systems, it does not scale to an organization as a whole. Every bigger organization is struggling with the fragmentation of information over multiple platforms, media and formats.

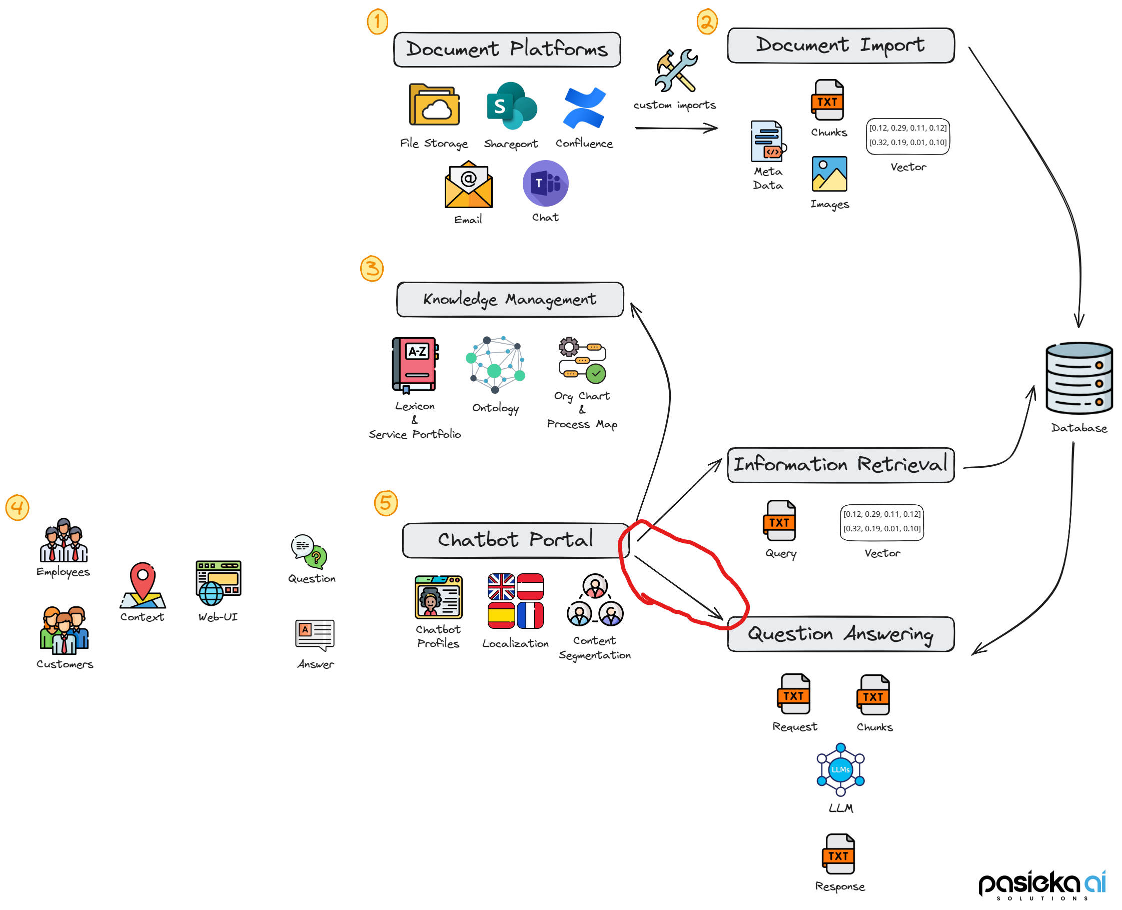
Research shows that modern knowledge workers spend up to half a day per week searching for information. Remote work and the ever increasing number of different communication platforms, only adds to the problem.

In this second Post, we will look at Enterprise Knowledge Systems, that can help organizations counter this trend and enable organization to provide their employees a single portal to access all information they need, quickly.

Type II - Enterprise Knowledge Systems

So far, we looked at Question Answering Systems for a limited domain using an common RAG workflow. Such a system is a good solution for use cases, where one wants to support the research work of an individual or a small team.

What we want to look at now, are **Enterprise Knowledge Systems (EKS)** that accelerate the information access and its use throughout an complete organisation and even its customers. Figure 2 illustrates such a system, highlighting its difference to Question Answering Systems.



The biggest difference compared to Q/A systems, is the scale and diversity of the information that is ingested by the system. Instead of individual documents, complete document platforms (1), like Confluence, Sharepoint, File Storage, or Email and Chat are being imported. This will in most cases, require custom import processes (2) that not only support custom file formats and different modalities, but most importantly, provide meta data about the imported documents. This meta data is stored along side the information extracted from documents (chunks and embedding vectors) and is vital during the information retrieval. It enables the enforcement of data access policies and increase the accuracy of the information retrieval by enabling more precise searches for relevant documents.

Importing a vast amount of information from different document platforms, creates a document corpus, that is not only hug in size, but in diversity of its content. When dealing with such big and heterogeneous document corpus, of overlapping domains and vocabulary, the information retrieval becomes a challenge. One way to cope with this challenge, is to perform explicit Knowledge Management (3) that tries to provide up-to date information about the organisation itself and its particularities. This includes information about company wording, products and services, org charts and in the best case an ontology, which is a formal representation of knowledge within the different domains, describing the relationship between concepts and entities.

This organisational information can be used during the import of documents to enrich the meta data, as well during the information retrieval to filter candidate documents. An ontology helps the system understand relationships between concepts, and a service portfolio allows for answering questions that connect documents to specific services and products.

While keeping the user interface to chat, similar to Q/A Systems, the information retrieval in EKS can be supported by including information about the user and her context (4) into queries. For example, in case of an employee, her position within the organisation and in particular their role can be taken into account in order to rank possible documents differently, or tailor the generated answers, to their profile.

All those efforts culminate in building an Chatbot Portal (5) that serves internal and external users as a single point of entry to access contextualized information from anywhere within the organization. Different audiences can be served by customized Chatbot Profiles that are implemented with RAG workflows that are backed by different data sources and system configurations (i.e. system prompts, personas and instructions). Based on the user context, the interaction with the system can localized and content segmented in a way to reflect existing organizational structures.

Example: How an Enterprise Knowledge Management System accelerates research

To give a concrete example, imagine a multi-national construction company that utilized a Knowledge Management System to support their employees.

Shiva, a french labour law expert, needs to create a report on ongoing construction projects in Germany. She utilizes the dedicated Project Inquiry Chatbot to get a description of all ongoing projects in Germany and details about them.

The KMS is able to ensure that Shiva only receives information that she is entitled to, excluding for example, budget and financial information. The system is not only able to base its answer on source documents in different languages, but in addition is able to leverage the company service portfolio and org chart to answer questions about the different services offered as part of the projects and the responsible project owners.

As a followup, she takes the list of project owners and uses another Chatbot, focused on Email and Chat conversations, to get a summary of previous conversations that she had with the different project owners and who she has not been in contact before.
Because she knows, how her German colleagues appreciate an Email in German, she uses the chatbot to formulate for each new contact a personalized Email, that explains her intent and the support she needs from them based on the particular project.

The KMS is able to assist Shiva because it has ingested information from different source systems, in this example, project reports from sharepoint and Emails. The KMS able to relate service and product names, and answer questions about them, that are mentioned in project reports with product documentation that is describe in the KMS Service Portfolio. The organizational chart helps to identify the project owners and their means of contact, that enables the Email Bot to identify relevant previous conversations.

Challenges with building and maintaining an KMS

A Enterprise Knowledge Management Platform is a powerful tool to accelerate a companies knowledge workers, but is not without its own challenges. Compared to RAG based Question Answering Systems, there are no out of the box solutions that one can deploy easily. The real benefit of a KMS depends on its deep integration with existing knowledge systems and documentation. It requires significant configuration, customization and maintenance work.

The biggest challenges include the following topics

- **Data Quality:** Unsurprisingly, the quality of the input data and the way it is ingested by the system, is essential to providing value. Similar to traditional challenges of Enterprise Search, an KMS will have difficulties to find relevant information in cases there are out-dated duplicates of documents. A KMS can support data hygiene by highlighting similar documents, but will rely on manual review and heuristics to avoid data duplication.
- **Content Segmentation:** The information retrieval over a big corpus remains the key challenge for an KMS. As part of the Knowledge Management, the information retrieval can be improved by segmenting the document corpus and reducing the candidate space for individual requests. This can happen explicitly by providing the user an option to filter source systems or domains that are being taking into account, or implicitly by taking the users context (i.e. role in an organization, previous requests, ...) and intent into account. If an ontology exists, that describes the different domains within an KMS, documents during ingestion can be tagged based on their content. This metadata about a document can then be used to filter during information retrieval based on the users intent.
- **Knowledge Management:** As described in the overview, the Knowledge Management aspect of an KMS helps the system to improve its Information Retrieval and the quality of the generated answers. This comes at the cost of resources that need to be invested to build the knowledge support structures and to maintain them. In addition there is the organisational question of who has the right and the responsible to maintain what part of the knowledge management system.
- **Continuously Increasing Request Complexity**: With an increasing data corpus and number of different Chatbot Profiles, the use cases and the number of edge cases increases. Requests tend to become more complex, as uses discover new possibilities to leverage the system for their needs. This includes chained multi-part questions, requests for information across multiple different source systems and modalities, or long conversation with increasing dependency on per conversation context. Other requests can require temporal reasoning over information from a specific time period, or questions that necessitate understanding implicit relationships across documents. As a result, the system needs continuous development and tuning, ranging from more advanced data extraction and pre-processing methods like Visual Language Models that improve the extracted of structured data from unstructured source documents, to the continuous tuning of system prompts and agent profiles.
- **Data access:** The added value of an KMS is proportional to the data it has access to. Big organizations have the challenge that information is spread across many platforms and technologies. Providing data access to the varying platforms can be an organizational challenge, specially because the established data access permissions have to be kept in tact. A KMS is a Chief Information Officers (CIO) dream and nightmare in one. Ensuring that information ingested by the KMS is only available with the appropriate permissions, is essential. An additional factor is the freshness of data available through the KMS. Depending on the size of Document platforms, and their technical capabilities, there might be a significant delay between the the creation of information in a source system and until it is available in the KMS.
- **ML Cybersecurity (MLSEC)**: As described about the challenges of Data access, a KMS is a particular interesting target for attackers, as a compromise of a centralize system of this kind, allows access to a huge amount of sensitive information. Besides traditional cybersecurity concerns, a KMS is susceptible to machine learning security issues like data poisoning and in particular indirect prompt injection (see this post), an attack vector that uses specially crafted documents that contain Jailbreak instructions for an LLM, manipulating the answers generated by the system in security compromising ways. An overview of LLM specific cybersecurity threads can be found as part of the OWASP Top 10 LLM risks.
- **Content moderation**: Lastly, when providing customers and external partners access to specific public facing Chatbots, organizations starts to care about moderating the answers generated by the KMS. Unfortunately, making sure, answers aligns with the organizations political and ethical guidelines, similar to Jailbreak, is an unsolved problem. Providing a tailored system prompt and Chatbot profile can get a long way, to nudge the system to behave, but one should always be aware, that malicious crafted user input can cause the system to generate answers that do not align with an organizations content policy. Models hosted by LLM service providers or cloud services like OpenAI, Anthropic, Azure, AWS, GCP, either have configurable content filters in place or provide moderation models that one can use to classify user or model generated content. For custom solutions, one can build on projects like deepval, guardrails.ai or Nvidia - NeMo Guardrails.

Recommendations for Implementing an KMS

There are a limited number of commercial Enterprise KMS systems available, like Guru or Squirro but as described previously, the value of an KMS is proportional to its integration and adaptation to an organization. Organizations have therefor the challenge to decide if they either invest in using a commercial KMS system and configure it to their demands, or invest in building their own solution.

Organization that can invest in supporting a development team of 5-7 people, can expect get returns on their investment within 12-18 months, and to be able to experiment with prototypes significantly earlier. Specially companies that already develop software that can benefit from text analysis, understanding and generation should evaluate to invest in building their own solution that serves to accelerate internal use cases, but can be the stepping stone to extending existing products and services, or develop new ones.

The following recommendations will hopefully be useful for such projects:

- **Split System Evaluation:** During development, but as well during operation of the KMS, evaluate Information Retrieval and its Question Answering separately. Separating the two, makes it easier to automatically evaluate changes to the system as well to detect drift in performance during operation. The evaluation of Information Retrieval can be automatized by Mean reciprocal rank (MRR) scoring of test data in data corpora of different size. Question answering performance, can be evaluated by comparing generated answers with ground truth answers using LLMs.
- **No single model to rule them all:** Plan from the beginning to be able to switch models rapidly, not only to be able to leverage the newest foundation models, but as well to benefit from the increasing performance of open source small language models. Enable the use of different models for varying stages of the input processing and answer generation and evaluation. Using a small but specialized model for information extraction like olmOCR, Docling or API solutions like Mistral OCR can improve data quality significantly while keeping data processing costs and time low. Using differently LLMs for the varying steps in Answer Generation, like answer ranking, groundedness and relevance checks can as well reduce costs and latency.
- **Beyond Python:** A KMS is a complex system with only small parts of it that require a deep understanding of NLP and LLMs. If your organization has experienced developers in system languages like Java, GO or Rust, build as much of the system in those languages and keep the python development to core components like data extraction and knowledge representation. More about this in a nother blog post of mine: Blog: The Pain of hiring Python AI Engineers
-

Where KMS fail

KMS still "only" support knowledge workers to find information faster and help with generating short texts and documents.

They are the wrong tool when trying to automatize multi-step reasoning tasks or use them through non-chat based interfaces.

For solving those tasks, one needs a more agentic approach, which lead us to the third and final type of knowledge-based systems, the AI Agent platforms.

Key Takeaways for Knowledge Management Systems

- The value an KMS provides depends on its integration with existing document platforms.
- When building your own solution, be aware of the size of the project and the unresolved challenges.
- Don't try to use KMS for process automation, use and AI Agent Platform instead.

Thank you for reading, I hope you found it useful. In the next post we will look at AI Agent platforms, that accelerate the work of knowledge workers by using AI agents to automatize pre-defined workflows.

Building LLM powered Knowledge-based Systems - from RAG to Multi-agent AI Platforms - Part 3

Hello and welcome back to this three part Series on Knowledge-Based Systems. In the second part, we introduced Enterprise Knowledge Management Systems that provide centralized access to information within an organization.

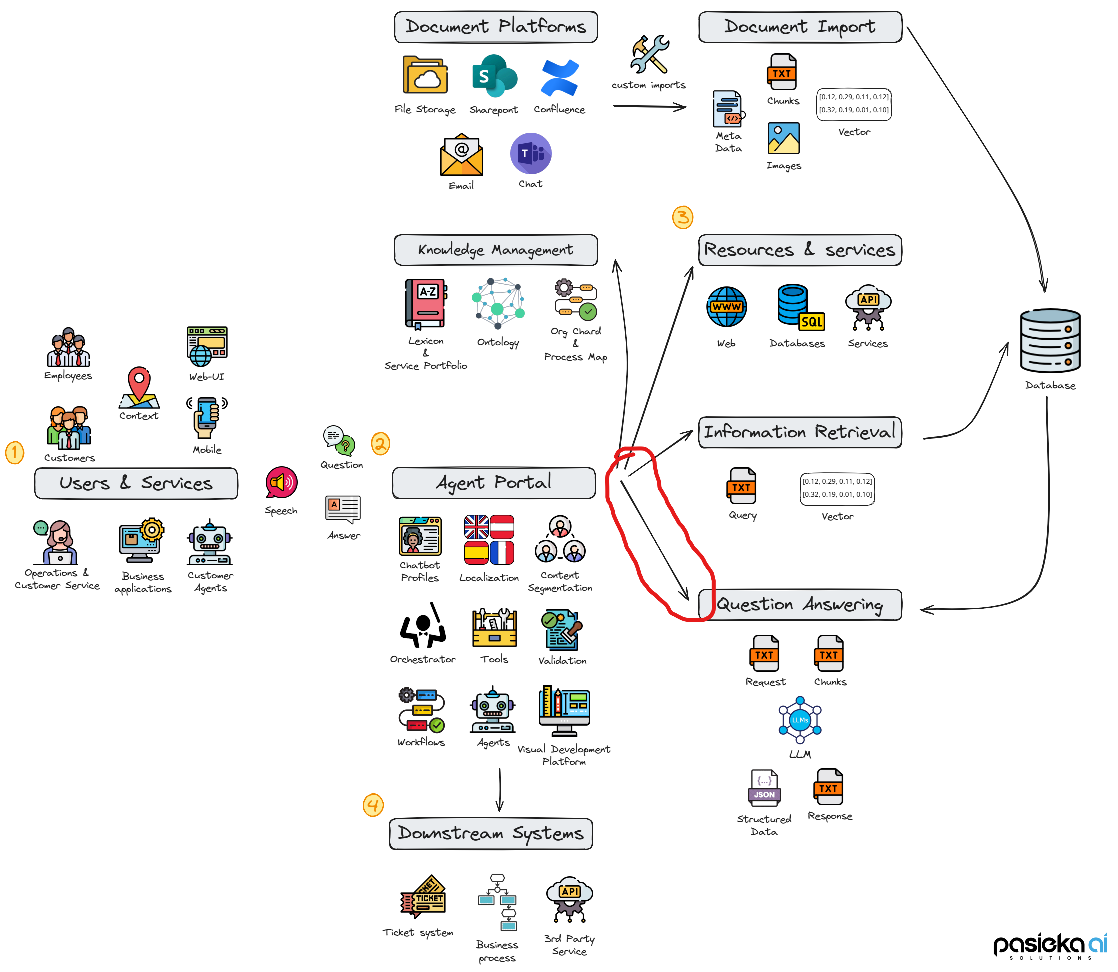
In this part, we will go further and not only accelerate the access to information, but will describe how LLM powered AI Agents can be used to automatize pre-define business processes.

Type III - AI Agent Platforms

AI Agents have been a very hot topic over the last months. Many see the future frontier of AI in developing autonomous AI Agents with super human capabilities. With use cases that include personal assistants, AI Software Developers, AI Researchers or robots operating in the physical world.

Those use cases are not the focus of this work. Instead I want to look at knowledge-based systems as AI Agent Platforms that are a lot closer to "traditional" Enterprise Robotic Process Automation (RPA) that came into flourish in the early 2000's with companies like UiPath Blue Prism and provide solutions to automate pre-defined business processes. Further, we look at AI Agent platforms, that make use of the available data corpus and service of an organization to automate and support knowledge work.

Figure 3 illustrates an AI Agent Platform and what additional capabilities it has compared to a Knowledge Management System.



In contrast to KMS systems, an AI Agent platform is not only serving human users through a chat interface, but is intended to be used by other services and AI Agents (1). To enable this an Agent Portal (2) is extending the Chatbot Portal of the KMS system to permit the development of Workflows that are executed by semi-autonomous AI Agents with the access to software tools and services. A separate Agent is taking the place of an orchestrator, that is responsible to identify the intent of a request and to coordinate how requests are routed to different agents and what workflows are triggered. Platform users can define their own workflows and configure agents using visual development platforms through no- or low-code options. The different Agents are augmented with the capability to use not only search the information ingested from different document platforms, like a KMS could, but in addition can access external resources and services through versatile API calls (3). This includes services like web search engines and can include in combination with tools the capability to automatically query external databases. While Agents can still serve use cases in which they respond with human readable answers in a chat interface, they capabilities are extended to be able to trigger themselves other "downstream" services (4). Such down stream services could be existing business processes or internal and external 3rd Party services.

Where the KMS has the focus on providing quick and easy access to an organizations information, plus limited generative capabilities, an AI Agent platform extends the capabilities of Agents to incorporate external resources and leverage external services in order to achieve pre-defined workflows.

A KMS is a tightly managed system that provides operators of the system to capability to control the data import process ensuring data quality and manage Chat Bots and Profiles that enable a specific set of use cases.
The AI Agent platform in contrast enables a more self-service like approach, where users can build their own workflows and agents that serve their particular use case. This introduces additional challenges, which will be discussed later, but provide a chance for a bigger audience to benefit from the platform and encourages a "organic" growth and exploration of new use cases.

Example: How an AI Agent platform transforms customer service

Imagine a customer service platform for an electronic consumer device manufacturer. The company has a AI Agent platform in place that supports human service operators by handling multiple steps in the customer service journey. The following is an illustration of such an customer journey.

The first touch point, is a customer service voice bot that is having a conversation with a customer reporting an issue with her smartphone speaker not being able to connect to her smart home system. The voice bot is following a partially pre-defined script, similar to how a human operator would work, while providing the customer the freedom to have a natural conversion. After failing to resolve the customers issue, the voice bot is creating a support ticket, including all relevant information contained in the conversation so far and inquiring about additional relevant topics like the smart home setup.

Another support ticket Agent is analyzing the newly created ticket and starts to augment the customer provided information with background information that can help to resolve the case. This includes for example, public available information about the products used in the smart home setup, acquired through web search. Based on the new extended support ticket, the Agent creates a suggestion on how to resolve the issue, relying on product documentation and previous support cases that describe similar issues. Part of the suggestions is a technical description of the likely causes of the issue, tailored to explain the situation clearly to the human customer service operator as well creating a draft for an email to the customer, explaining how to resolve the issue.

When the human service operator open the support case, they find all information they need to verify if the diagnosis of the AI Agent is correct and if they agree with the proposed solution. They focus on validation and evaluation of the AI Agents work, instead of using tools, like an KMS to find a solution themselves.

When the human operator approves the proposal, the Ticket AI Agent takes over again, in order to update the ticket and sends an email to the customer.

Example: How an AI Agent platform enables employee driven use case development

The following example shows, how User created workflows, covering new use cases, when growing in popularity can become an part of the standard platform functionality to benefit a bigger audience.

André as a team lead in a medium sized company is having many online meetings for which he tries to make sure that every meeting provides valuable outcomes and clear to-dos.

Keeping track of all to-dos that come up in a meeting can be cumbersome and adding them to a Issue tracking system even more sore.

For that reason André has created a short workflow on the companies AI Agent platform. To use it, he adds a bot to an online meeting. Once the meeting has ended, the Agent workflow is triggered. As part of the workflow, the meeting transcripts are analyzed (if transcripts do not exist, they are first created with a Speech-to-Text model) and a list of Todos including who is responsible and optional deadlines are created. Based on the list, an Agent is creating separat Todo tickets, including a short description, deadline and a summary of the context in which the task was defined.

Other employees within the company quickly see how the workflow helps André to save time and create their own copy. The Agent Platform team, that is continuously observing how people are using the platform, can see the opportunity to help pople within the company and create a slightly modified and more resilient Todo Agent, that Platform users can make use without creating and maintaining their own.

Current Landscape of AI Agent Platform

AI Agent and Agentic Systems in general, have received a lot of attention in the last months, pushing forward the hype train led by LLMs. Many companies providing chat bot and robotic process automation solutions, have pivoted towards AI Agent platforms (see Zapier Agents, Botpress, Make.com, ChatBase). The market is moving so quickly at the moment, that it is difficult to predict with precision, where its heading and to pick favorites.

Broadly speaking, there are three category of AI Agent Platforms and Frameworks at the moment.

- **Business Process Automation**: That focus on enterprise integration with existing data sources and services, providing no- or low-code platforms. Example of this are Stack AI and Microsoft Copilot Studio.
- **General Purpose AI Automation**: Targeted towards a bit more technical audience then the Business automation platforms, with great flexibility to implement a vast number of automation use cases ranging from individual so small business. Examples of this are n8n or relevance AI.
- **Developer Frameworks**: Software Frameworks, heavily python focused, that provide building blocks to implement custom AI Agent systems. Examples of this are Microsoft - AutoGen, LangGraph, PydanticAI or OpenAI Agents.

Having so many options on the table, the question arises, which one to choose under what circumstances. When to buy and when to build.

First of all, because of the velocity with which the technology is evolving and the rate of new products and frameworks emerging on the market, it is essential to prevent any vendor and platform lock-in, as well as to rely heavily on a single technology or solution.

The big majority of companies will benefit from AI automation, and can do so, without the need to be on the bleeding edge.

Depending on your sector, the investment into AI automation will be more or less pressing, but I would highly advice to gather experience and develop use cases by performing small prototypes using either general purpose AI automation platforms or Business Process Automation systems before thinking about any bigger investment. Like the underlying LLMs, the platforms develop rapidly and current limitations may very well disappear within the next 6 months, making it possible and useful to re-evaluate the feasibility of previously failing use cases on a regular bases.

Building your own AI Agent platform

Building your custom AI Agent platform is a major endeavour with unclear benefits over the quickly evolving AI Agent platform market. It requires a dedicated team made up of experts in cloud/on-prem infrastructure, system developers, AI experts and UX developers and at least 12 months to get a first working platform.

Building a platform will require to solve all challenges that have been presented for Knowledge Management Systems, plus the following

- **Unstable Ecosystem:** The previously mentioned Developer Frameworks are all very new and building on them will likely require re-writes in the future. I think that many of even the highly popular frameworks like Langchain are great to get experiments started, but provide a lot of weight that is not needed for each particular project. Getting inspired by the different frameworks, and implementing the required parts for your own project is a viable choice. As mentioned in this blog post, I see great value in limiting the use of python to experimentation and essential AI components of the system that require the python ecosystem and tooling. Choosing therefor a microservice architecture that provides the flexibility to implement different services in different technology stacks is essential.
- **Observability / Agentops**: The stochastic nature of LLMs and therefor AI Agents that build on them, require continuous monitoring and logging in order to study unintended system behavior. Multiple frameworks have emerged that focus on AI Agent use cases, like Langfuse , OpenLIT or Agentops. Similar to the Developer Frameworks mentioned previously, AI agent specific observability tools might have certain convenience, but when building a project of this size, I would recommend to rely on more established observability technologies like Open Telemetry, specially because they as well try to improve their support for AI agent use cases as discussed here.
- **Interoperability between Systems**: AI Agents are empowered by the tools and services they can use. Writing and maintaining custom adapters to external resources and services is very resource consuming. Although Anthropic has gathered certain momentum with its Model Context Protocol that could serve as a standard that helps to connect Agents to external services, it is not yet widely used and its at the moment unclear if its adaption will continue. For the near future it is therefore very likely that one needs to develop and maintain a high number of adapters.
- **Agent to Agent Communication**: AI Agent platforms encourage Agent-to-Agent communication in order to extend the capabilities of individual agents. This includes agents on the same platform, but as well external agents that are available as API services. At the moment no standard exists and the Model Context Protocol is not explicitly covering this use case. What happens in practice, is that at the moment one model is communicating with another one through human language, including all its inefficiencies and potential for misunderstanding. This increases the brittleness of the system as a whole, as the behavior of both agents is stochastic in nature on an individual level and established best practices like prompt engineering to dictate the behavior of a model are often model dependent. Meaning that optimizing the prompt of a model A to communicate with a model B in certain way, suddenly has to attend the model peculiarities of two models, instead of one. In case of both models running on the same platform, under one organisation, this might be feasible, but as soon as an Agent starts to communicate with Agents on other platforms that have the freedom to change the underlying behavior at any time, the systems stability is at risk.
- **Agent & Service discovery**: In distributed systems, there is the well known challenge of service discovery, that solves the problem to identify what services and resources are available, where. In case of a growing AI Agent platforms, where internal and external Agents, tools and services are coming together, a similar discovery service is needed.

Where AI Agent Platforms fail

As described in the introduction of this part, the AI Agent platforms discussed here, are not to be confused with the Autonomous Agents that solve multi-step reasoning tasks, but expecting exactly that from such a platform will set it up for failure.

The AI Agents are capable to detect requests intents and map them to existing workflows and tools. Based on modern LLM's, Agents are capable to understand and generate human language, that enables sophisticated text extraction and transformation use cases. Armed with access to tools and services, Agents can be used in complex workflows that solve many common knowledge worker tasks, or support them, so that human users can focus on validation and decision making.

AI Agents Platforms can accelerate business processes, help employees to increase their efficiency and automatize narrow work flows, but are not capable of replacing an employee.

AI Agent Platforms are a natural progression of digitization and automation that will broaden the responsibilities and capabilities of modern knowledge workers.

Key Takeaways for AI Agent systems

- Many different AI Agent platforms exist and are being developed rapidly.
- Avoid picking favorites or building your own, instead invest in experimenting with specific use case.
- When building your own solution, be aware of the many open problems related to Knowledge Management Systems and AI Agent platforms.

The present and future of AI Agent Systems

In this last part, I want to reflect and hypothesis on what the current limits and possible future frontiers for AI Agents are.
I distinguish therefore, between three type of AI Agents that share commonalities and might in the future even become one, but currently are still distinct.

- **Assistants**: are Systems that are used by humans as a tool or copilots. Either through text or voice, the human is "talking" to the assistant, asking it questions or giving it instructions. Most assistants we see so far are tightly integrated into a specific system and domain, to limit its scope and improve its results. This are currently the most deployed type of AI Agents for use cases like information retrieval, content creation, simple customer support and many more. They excel in use cases where the result of the assistants work is easy for humans to evaluate and failure causes little harm. Judging a picture or a piece of music, created with the help of an assistant is easy and if the assistant fails, nothing is lost. On the other hand, using an assistant for tasks like planning and booking a short trip is more complicated, because checking the correctness of all individual bookings and if they align with your preference takes a lot of time and in case of a problem, resolving it might be time consuming or costly. If the system is so reliable and capable of performing all requests correctly, we get into the field of autonomous agents.
- **Autonomous Agents**: are capable to without human supervision solve multi-step reasoning problems, like the development of complex software systems, manage a company, perform research or even operate in the real world. Many of the limitations and challenges of the varying knowledge based systems, could be resolved with such autonomous agents. Many see them equivalent to Artificial General Intelligence (AGI) and a cornerstone in modern AI Research in academy and private industry. As it stands currently, the limiting factor of these systems is the high failure rate in multi-step reasoning chains. If an Agent has to perform 10 steps to solve a problem, even with a success rate of 95% in each step, will only reach its goal in 59% of the cases. As with many challenging tasks, specially in the real world, and this includes the real digital world, the verification of a result is a problem that AI Agents struggle with. Failing to judge if the result of an action is positive and to what extend it aligns with a goal is one of the major blockers of Autonomous Agents. For a discussion of current technologies have a look at my blog post reviewing the biggest AI trends of 2024.
- **Delegates**: are what I call AI agent that serve as representative or interface between organizations and individuals. An example of a delegate agent, is one in which for example the human resources department is maintaining an "HR Agent" on an AI Platform that can be used by employees to book holidays, ask contract related questions or in general handle support requests in place of a human work in HR. Instead of answering requests directly, the HR team is maintaining "their" own HR Agent that is handling the majority of requests for them. This combines the scalability of Agents with the flexibility of human knowledge workers and ensures that the agent provides qualitative results and stays relevant. In bigger organizations this enables "local" control while enabling global synergies through an agent-to-agent communications and work flows that combine multiple delegates from different teams.

Assistants are already quite prominent and I expect specially their adoption to be used through voice to accelerate on mobile devices.
With the adoption of AI Agent platforms, we will see a rise in Delegate Agents and I expect certain roles like first level customer support to be replaced with Delegates Agents in the near future.

With respect to Autonomous Agents, its very difficult to make predictions. Systems like Manus try to bridge the gap between Assistants and Autonomous Agents by doing their best to improve the reliability of the system to improve its success rate and reasoning capabilities, but more than that, by providing a user experience which enables the human operator to understand and be aware of what the system is doing and by eliciting human feedback and supervision while giving the human the impression that the Agent is doing most of the work.

Conclusion

We have seen three types of knowledge based systems:

- **Question / Answering Systems** that help an individual or team to gather information faster.
- **Knowledge Management Systems** that provide organizations a single point of access to all their otherwise distributed knowledge.
- **AI Agent Platforms** as the means to automatize business workflows.

For each type we discussed the biggest challenges when building and operating them.

Ending with a look at the current and future application of AI agents as:

- **Assistants** that help humans with specific tasks.
- **Autonomous Agents** that without human supervision solve multi-step reasoning exercises.
- **Delegates** that teams maintain to perform work on their behalf.

That's all Folks!