Contrato*360* 2.0: A Document and Database-Driven Question-Answer System Using Large Language Models and Agents

Antony Seabra[1,2][a], Claudio Cavalcante[1,2][b], João Nepomuceno[1][c], Lucas Lago[1][d], Nicolaas Ruberg[1][e] and Sergio Lifschitz[2][f]

[1] *BNDES, Área de Tecnologia da Informação, Rio de Janeiro, Brazil*

[2] *PUC-Rio, Departamento de Informática, Rio de Janeiro, Brazil*

Keywords: Information Retrieval, Question Answer, Large Language Models, Documents, Databases, Prompt Engineering, Retrieval Augmented Generation, Text-to-SQL.

Abstract: We present a question-and-answer (Q&A) application designed to support the contract management process by leveraging combined information from contract documents (PDFs) and data retrieved from contract management systems (database). This data is processed by a large language model (LLM) to provide precise and relevant answers. The accuracy of these responses is further enhanced through the use of Retrieval-Augmented Generation (RAG), text-to-SQL techniques, and agents that dynamically orchestrate the workflow. These techniques eliminate the need to retrain the language model. Additionally, we employed Prompt Engineering to fine-tune the focus of responses. Our findings demonstrate that this multi-agent orchestration and combination of techniques significantly improve the relevance and accuracy of the answers, offering a promising direction for future information systems.

1 INTRODUCTION

Contract management in large corporations involves overseeing legally binding agreements from their initiation through to execution and finalization. This process encompasses ensuring that services or products are delivered in accordance with contractual terms, monitoring their execution, and continuously evaluating both operational and financial performance throughout the service or product lifecycle. In the case of public sector companies, this process becomes even more complex due to stringent regulatory frameworks. In Brazil, for instance, Law No. 14,133/2021 mandates that contract management includes a wide range of activities, such as technical and administrative oversight, adherence to contract duration, reevaluation of economic and financial terms, modifications to service scope, and the enforcement of penalties and fines when necessary. These regulations impose an additional layer of complexity on the contract management process, demanding a robust and systematic approach to ensure compliance and efficiency.

Beyond contract managers, dedicated organizational units are essential to support the contract management process, ensuring that the diverse range of activities associated with contract execution is managed efficiently. Often, these units require specialized knowledge to handle complex services effectively. Notable examples include information and communication technology (ICT) services, property and asset management, and construction and engineering projects, each of which demands a high level of expertise. Additionally, these units typically rely on Contract Management Systems (CMS) to streamline their operations. Public companies may either develop these systems in-house or opt for widely-used market solutions, such as SAP Contract Life-cycle Management and IBM Emptoris Contract Management, among others.

While these systems efficiently handle general contract information, such as signatures, expiration dates, payment terms, and contract agents, many specific details required to support effective management activities remain accessible only through the original documents. For instance, traditional Contract Management Systems (CMS) are often unable to respond to inquiries concerning particular aspects of a contract, such as penalties, discounts, or fines associated with delays in service or product delivery. More-

[a] https://orcid.org/0009-0007-9459-8216
[b] https://orcid.org/0009-0007-6327-4083
[c] https://orcid.org/0009-0004-5441-8426
[d] https://orcid.org/0009-0001-4094-1978
[e] https://orcid.org/0009-0005-4388-4656
[f] https://orcid.org/0000-0003-3073-3734

Seabra, A., Cavalcante, C., Nepomuceno, J., Lago, L., Ruberg, N. and Lifschitz, S.
Contrato360 2.0: A Document and Database-Driven Question-Answer System Using Large Language Models and Agents.
DOI: 10.5220/0013070400003838
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 16th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2024) - Volume 1: KDIR*, pages 167-178
ISBN: 978-989-758-716-0; ISSN: 2184-3228
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

over, they lack the capability to provide insights into comparative characteristics across different contracts, such as penalty clauses related to database support agreements. These tasks are highly time-consuming.

The objective of this study is to provide a solution that aids contract managers in addressing queries related to both contract documents and data housed within traditional Contract Management Systems. One of the key challenges faced by contract managers is the time-consuming process of searching for and retrieving relevant information from lengthy and complex contract texts. To address this, we leverage state-of-the-art large-scale Language Modeling (LLM) technologies to analyze and extract pertinent details from contract documents efficiently. This not only improves the accuracy of the information retrieved but also significantly enhances the productivity of contract managers by reducing the manual effort required to locate specific information. Additionally, our approach integrates data from traditional Contract Management Systems, ensuring that responses are both relevant and comprehensive, thereby streamlining contract management activities.

In this work, we evaluated and integrated several Natural Language Processing (NLP) techniques to develop a Q&A system specifically designed for IC contracts, using contract PDF files and data from Contract Management Systems (CMS) as primary data sources. To enhance the relevance of user queries, prior work by (Seabra et al., 2024) employed Retrieval-Augmented Generation (RAG) techniques and a static approach to text-to-SQL for extracting relevant metadata from contract systems. Building upon this, our approach utilizes agents to dynamically improve the accuracy and contextual relevance of responses, with a particular focus on a context-aware text-to-SQL agent that interprets user queries more effectively. Furthermore, similar to (Seabra et al., 2024), we applied Prompt Engineering techniques to standardize responses and ensure greater precision in the answers provided.

One of the primary challenges in interpreting contract documents lies in distinguishing between relevance and similarity, a complexity that arises due to the standardized formats and repetitive textual structures commonly found in these documents. This standardization is a challenge for LLMs because there is a great deal of textual similarity, which does not necessarily translate into relevance. Using a mix of NLP techniques, we developed a solution that minimizes the impact of standardization and provides relevant answers. This approach made it possible to design a solution without needing traditional *fine-tuning* or re-training of language models.

The paper is organized as follows: Section 2 provides technical background on LLMs, *RAGs text-to-SQL*, *agents*, and *prompt* engineering. Section 3 discusses the methodology of the use of the presented techniques, while Section 4 details the architecture of our solution. Section 5 describes how we evaluated the proposed solution and the experimentation of the Q&A application. Finally, Section 6 concludes our study and proposes directions for future research in this field.

2 BACKGROUND

The dissemination of several applications in the area of Natural Language Processing (NLP) was made possible by Large Scale Language Models (LLMs), including question and answer (Q&A) systems. Recently, the use of agents has been introduced as a crucial component in LLM-based systems to orchestrate and manage task execution dynamically. Agents, such as router agents, SQL agents, and RAG agents, enable the efficient allocation of tasks by directing queries to the most suitable processing modules, enhancing system adaptability and performance. This approach allows LLMs to better handle complex queries, making responses more accurate and contextually relevant by integrating external data sources and specialized processing routines (Mialon et al., 2023).

2.1 Large Language Models

Large-scale Language Models (LLMs) have revolutionized the field of natural language processing with their ability to understand and generate human-like text. In their architecture, they utilize a specific neural network structure, *Transformers*, which allows the model to weight the influence of different parts of the input texts at different times (Vaswani et al., 2017).

Conversational applications, a specific use case for LLMs, specialize in generating text that is coherent and contextualized. This is achieved through training, in which the models are fed vast amounts of conversational data, allowing them to learn the nuances of dialogue (OpenAI, 2023a). In this way, LLMs have established a new paradigm for NLP. Moreover, by expanding the search space with external data or specializing through fine-tuning, LLMs become platforms for building specialized applications. In this work, all language models utilized were based on OpenAI's GPT series. Specifically, we employed the *text-davinci-002* model for generating embeddings and the *gpt-4-turbo* model for generating answers to user queries.

2.2 Retrieval-Augmented Generation (RAG)

According to (Chen et al., 2024), LLMs face significant challenges such as factual hallucination, outdated knowledge, and lack of domain-specific *expertise*. In response to these challenges, RAG represents a paradigm shift in the way LLMs process and generate text. The principle behind RAG involves using vector storage to retrieve text fragments similar to the input query (Gao et al., 2023b). This technique converts both the query text and the information database into high-dimensional vectors, allowing one to retrieve similar information, which is then fed to an LLM.

(Gao et al., 2023b) and (Feng et al., 2024) describe *frameworks* that exploit the advantages of this technique by providing additional data to the LLM without re-training the (Li et al., 2022) model. By dividing the available text into manageable chunks and embedding these chunks in high-dimensional vector spaces, it is possible to quickly retrieve contextually relevant information in response to a query, which informs the next processing steps. As shown in Figure 1, the first step (1) involves reading the textual content of the PDF documents into manageable chunks (*chunks*), which are then transformed (*embedding*) (2) into high-dimensional vectors. The text in vector format captures the semantic properties of the text, a format that can have 1536 dimensions.

These *embeddings* vectors are stored in a *vectorstore* (3), a database specialized in high-dimensional vectors. The vector store allows efficient querying of vectors through their similarities, using the distance for comparison (whether *Manhatan*, Euclidean or cosine).

Once the similarity metric is established, the query is *embedded* in the same vector space (4); this allows a direct comparison between the vectorized query and the vectors of the stored chunks, retrieving the most similar chunks (5), which are then transparently integrated into the LLM context to generate a *prompt* (6). The *prompt* is then composed of the question, the texts retrieved from the *vectorstore*, the specific instructions and, optionally, the *chat* history, all sent to the LLM which generates the final response (7).

In RAG, the *chunking* strategy is important because it directly influences the quality of the retrieved information. A well-designed chunk generation ensures that the information is cohesive and semantically complete, capturing its essence.

A key aspect of RAG is the difference between similarity and relevance. Similar passages may not contain the information relevant to answering a query, posing a challenge to accurately retrieve information, especially in cases where data comes from multiple documents with similar structure. In such contexts, documents may share a high degree of structural and lexical similarity, making it difficult for retrieval algorithms to distinguish between content that is merely similar in form and content that is truly relevant to a query.

2.3 Text-to-SQL

Text-to-SQL is a technology that enables the conversion of natural language queries into SQL commands based solely on the database schema, eliminating the need for knowledge of the underlying data (Liu et al., 2023). This approach leverages the capabilities of LLMs to understand and interpret human language, allowing users to retrieve data from databases through plain text input without requiring specialized knowledge of SQL syntax (Gao et al., 2023a).

By translating natural language into SQL queries, *text-to-SQL* brings complex database structures and end users closer together, making access more intuitive and efficient. This technique is particularly useful because it allows non-expert users to access databases by asking natural language queries. It improves data accessibility, reduces the learning curve associated with database querying, and speeds up data analysis processes, enabling more users to make data-driven decisions.

The main distinction between RAG and *text-to-SQL* techniques lies in how information is retrieved. RAG relies on retrieving text segments from a vector store that are similar to the user's question, and using these segments to generate a coherent and contextually relevant answer. This method is effective for questions where the answer can be synthesized from existing text. However, it is not always possible to identify the information expected as the answer. In another aspect, *text-to-SQL* translates natural language queries into SQL commands, as demonstrated in (Pinheiro et al., 2023), which are then executed against a structured database to retrieve exact data matches. This ensures that if the text-to-SQL translation is accurate, the user will receive a highly specific answer directly from the database fields.

Therefore, while RAG operates on the principle of textual similarity and generative capabilities, *text-to-SQL* offers a more intrusive mechanism for data retrieval by executing queries that directly match the user's intent, making it particularly effective for data investigations.

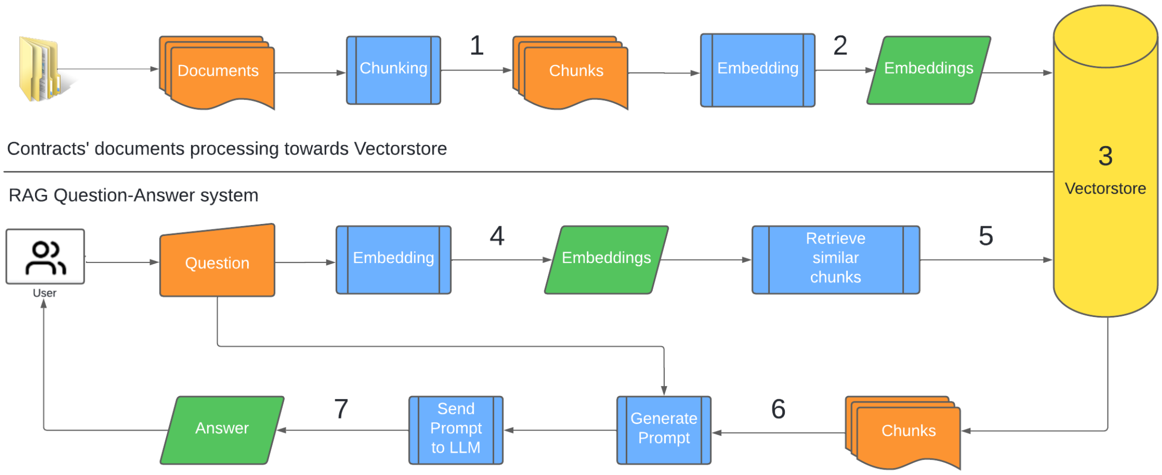


Figure 1: Retrieval-Augmented Generation.

2.4 Prompt Engineering

Prompt engineering is the art of designing and optimizing *prompts* to guide LLMs in generating desired outputs. The goal of *prompt* engineering is to maximize the potential of LLMs by providing them with instructions and context (OpenAI, 2023b).

In the context of *prompt* engineering, prompts are a fundamental part of the process. Through prompts, engineers can outline the script for a response, specifying the desired style and format for the LLM response (White et al., 2023) (Giray, 2023). For example, to define the style of a conversation, a *prompt* could be formulated as ”Use professional language and treat the customer with respect” or ”Use informal language and emojis to convey a friendly tone.” To specify the format of dates in responses, a *prompt* instruction could be ”Use the American format, MM/DD/YYYY, for all dates.”

On the other hand, context refers to the information provided to LLMs along with the main prompts. The most important aspect of context is that it can provide additional information to support the response given by the LLM, which is very useful when implementing Q&A systems. This supplemental context can include relevant background details, specific examples, and even previous dialogue exchanges, which collectively help the model generate more accurate, detailed, and contextually appropriate responses. According to (Wang et al., 2023), *prompts* provide guidance to ensure that the model generates responses that are aligned with the user's intent. As a result, well-crafted *prompts* significantly improve the effectiveness and appropriateness of responses.

Recent studies have begun to explore the synergistic integration of these techniques with LLMs to create more sophisticated Q&A systems. For example, (Jeong, 2023) reinforces the importance of using Prompt Engineering with RAG to improve the retrieval of relevant documents, which are then used to generate both contextually relevant and information-rich answers. Similarly, (Gao et al., 2023a) explores the integration of *text-to-SQL* with Prompt Engineering to enhance the model's ability to interact directly with relational databases, thereby expanding the scope of queries that can be answered accurately.

2.5 Agents

The use of agents in applications built around Large Language Models (LLMs) is relatively recent but has already became common. Agents act as intelligent intermediaries that route, process, and present information in ways tailored to the context of the query. These agents leverage recent advancements in AI, such as Retrieval-Augmented Generation (RAG) and tool utilization, to perform more complex and contextually aware tasks (Lewis et al., 2020). They play a pivotal role in orchestrating complex tasks, integrating various data sources, and ensuring that the system responds accurately and efficiently to user queries.

In a complex LLM-based system, different tasks often require specialized handling. Agents enable task orchestration by directing queries to the most appropriate component, whether it's for retrieving data, performing calculations, or generating visualizations. For example, an application may have a Text-to-SQL agent to perform queries over a relational database and a Graph agent to visualize graphs after an answer, if appropriate. According to (Jin et al., 2024), applying LLMs to text-to-database management and query optimization is also a novel research direction in natural language to code generation task. By converting natural language queries into SQL statements, LLMs help developers quickly generate efficient database query code. In the realm of integrating heterogeneous data sources, Q&A applications often need to access data from documents, databases, APIs, and other repositories. Agents facilitate the seamless integration of these heterogeneous data sources, allowing the system to extract relevant information dynamically.

There are several agent types. As outlined in (Singh et al., 2024), agent workflows allow LLMs to operate more dynamically by incorporating specialized agents that manage task routing, execution, and optimization. These agents serve as intelligent intermediaries, directing specific tasks—such as data retrieval, reasoning, or response generation—to the most suitable components within the system. One of the most important ones in place are the Router Agents, as they are the decision-makers of the system. When a user poses a query, the router agent analyzes the input and decides the best path forward. For instance, if a query is identified as needing factual data, the router agent might direct it to a RAG model. If the question involves specific data retrieval from a database, it will engage an SQL agent instead.

As mentioned before, RAG and SQL Agents are very relevant too. According to (Saeed et al., 2023), SQL agents can effectively manage data retrieval tasks by leveraging LLMs. The SQL queries are transformed into prompts for LLMs, allowing the system to interact with unstructured data stored in the model, mimicking traditional database operations. (Fan et al., 2024) provides a comprehensive overview of the integration of RAG techniques in LLMs but moreover, (Wang et al., 2024) introduces a novel approach that combines RAG techniques with a drafting-verification process to improve the reasoning capabilities of LLMs when handling retrieved documents. The RAG agent, termed the ”drafter,” generates multiple answer drafts based on retrieved results, while a larger generalist LLM, the ”verifier,” assesses these drafts and selects the most accurate one. This approach effectively integrates retrieval and generation, enhancing the overall performance of LLMs in knowledge-intensive tasks such as question answering and information retrieval systems.

3 METHODOLOGY

To address the challenges faced by contract managers in terms of complex information retrieval, we propose Contrato360, a Q&A system supported by an LLM and orchestrated by agents. The system employs a range of techniques designed to enhance the relevance of responses while mitigating the risks associated with the standardized textual structures of contracts.

To achieve this goal of increasing the relevance of the responses obtained by Contrato*360*, we combined four techniques: 1) Retrieval-Augmented Generation (RAG) to increase the relevance of information about contracts contained in PDF documents; 2) Agents to orchestrate and route the flow of execution, enabling the dynamic selection of the most appropriate approach for each query context; 3) Text-to-SQL agent to retrieve the relevant information from contract systems; 4) Prompt Engineering techniques to standardize and ensure greater accuracy in the responses produced.

3.1 Applying RAG

One of the first decisions to be made is to choose the best strategy to segment the document, that is, how to perform the *chunking* of the PDF files. A common *chunking* strategy involves segmenting documents based on a specific number of *tokens* and an overlap (*overlap*). This is useful when dealing with sequential texts where it is important to maintain the continuity of the context between the *chunks*.

Contracts have a standardized textual structure, organized into contractual sections. Therefore, sections with the same numbering or in the same vicinity describe the same contractual aspect, that is, they have similar semantics. For example, in the first section of contract documents, we always find the object of the contract. In this scenario, we can assume that the best *chunking* strategy is to separate the *chunks* by section of the document. In this case, the *overlap* between the *chunks* occurs by section, since the questions will be answered by information contained in the section itself or in previous or subsequent sections. For the contract page in the example in Figure 3, we would have a *chunk* for the section on the object of the contract, another *chunk* for the section on the term of the contract, that is, a *chunk* for each clause of the contract and its surroundings. This approach ensures that each snippet represents a semantic unit, making retrievals more accurate and aligned with queries.

Having the contract section as the limit of the *chunks* improves the relevance of the responses within a single contract. However, when increasing the number of contracts that the Contract*360* intends to respond to, we observe the problem in correctly determining the contract to be treated. In the following example, we detail this aspect:

Consider the contract documents shown in Figure 3. showcases two service contracts be-

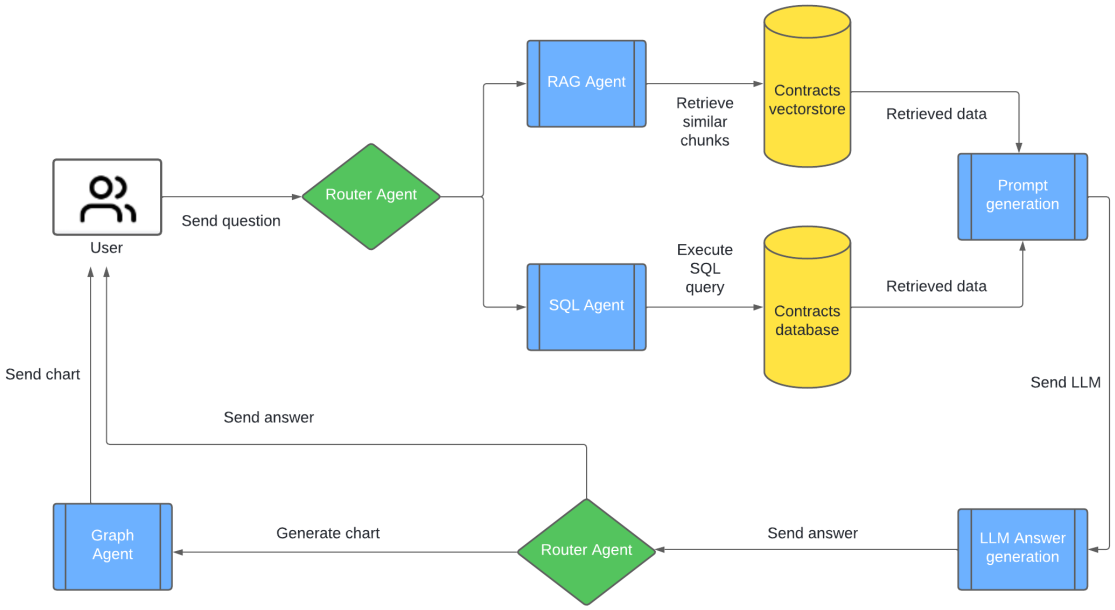


Figure 2: Methodology Workflow Combining Different Techniques.

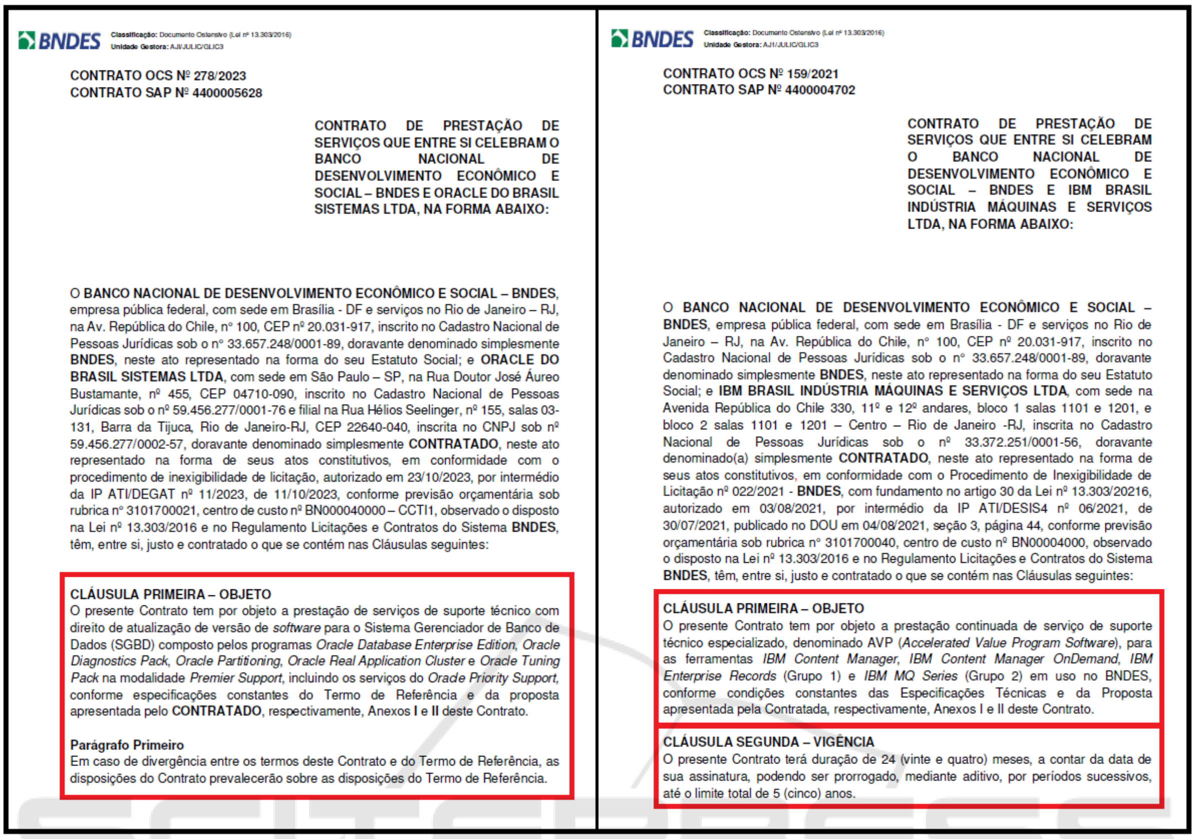
tween BNDES (Banco Nacional de Desenvolvimento Econômico e Social) and companies (Oracle do Brasil Sistemas Ltda. and IBM Brasil Indústria Máquinas e Serviços Ltda.), highlighting key clauses relevant to the provision of technical support and software updates. The contracts are presented in Portuguese, reflecting the original legal terms and specific obligations of each party. For instance, the contract with Oracle (Contract No. 278/2023) details the provision of services for Oracle Database and associated technologies, emphasizing software support and entitlement to updates. Similarly, the contract with IBM (Contract No. 159/2021) focuses on support services related to IBM Content Management software. The `"CLÁUSULA PRIMEIRA - OBJETO"` (first clause - object) details the object of the contract and a frequently asked question is: *"What is the object of contract OCS 278/2023?"*. In this example, the RAG will store vectors containing the sections of both contracts, since this clause is common to both. However, when we inspect what is expressed in the *chunk*, its content does not contain the contract number, Figure 3. Thus, with great probability, a query about a specific contract may return a segment (*chunk*) unrelated to the contract, for example OCS 159/2021, being retrieved instead of the contract we want. In the case of our example, the *chunk* referring to the question that should be returned is related to contract OCS 278/2023.

To overcome this issue, it is necessary to add semantics to the *chunks*, by including document metadata. And when accessing the *vectorstore*, use this metadata to filter the information returned. In this way, we improve the relevance of the retrieved texts. Figure 4 displays the most relevant metadata for the contracts (source, contract and clause). Where source is the name of the contract PDF file), contract is the OCS number and clause is the section title. Thus, for the question *"What is the object of contract OCS 278/2023?"*, the *chunks* of contract OCS 278/2023 are retrieved and then the similarity calculation is applied, retrieving the text segments to be sent to the LLM.

3.2 Applying *Text-to-SQL*

Contracts are dynamic and undergo several events like operational changes and management adjustments throughout their life-cycle. To deal with this complexity, organizations use contract monitoring systems, such as *SAP Contract Life-cycle Management* and *IBM Emptoris Contract Management*. These systems control several aspects, such as the technical person responsible for the contract, changes in the contractor's representative, and the end of the provision of services. During the contract term, these events can occur and significantly affect contract management.

The Contract*360* retrieves those events from the Contract Management System (CMS) and incorporates them so the LLM can provide relevant responses to the user. Therefore, a *text-to-SQL* technique was natural to implement the reasoning and decision-making task (Yao et al., 2023) to obtain relevant responses from the CMS database to the contract managers.



Classificação: Documento Ostensivo (Lei nº 13.303/2016)
Unidade Gestora: AJ/JULIC/GLIC3

CONTRATO OCS Nº 278/2023
CONTRATO SAP Nº 4400005628

CONTRATO DE PRESTAÇÃO DE SERVIÇOS QUE ENTRE SI CELEBRAM O BANCO NACIONAL DE DESENVOLVIMENTO ECONÔMICO E SOCIAL – BNDES E ORACLE DO BRASIL SISTEMAS LTDA, NA FORMA ABAIXO:

O **BANCO NACIONAL DE DESENVOLVIMENTO ECONÔMICO E SOCIAL – BNDES**, empresa pública federal, com sede em Brasília - DF e serviços no Rio de Janeiro – RJ, na Av. República do Chile, nº 100, CEP nº 20.031-917, inscrito no Cadastro Nacional de Pessoas Jurídicas sob o nº 33.657.248/0001-89, doravante denominado simplesmente **BNDES**, neste ato representado na forma do seu Estatuto Social; e **ORACLE DO BRASIL SISTEMAS LTDA**, com sede em São Paulo – SP, na Rua Doutor José Áureo Bustamante, nº 455, CEP 04710-090, inscrito no Cadastro Nacional de Pessoas Jurídicas sob o nº 59.456.277/0001-76 e filial na Rua Hélios Seelinger, nº 155, salas 03-131, Barra da Tijuca, Rio de Janeiro-RJ, CEP 22640-040, inscrita no CNPJ sob nº 59.456.277/0002-57, doravante denominado simplesmente **CONTRATADO**, neste ato representado na forma de seus atos constitutivos, em conformidade com o procedimento de inexigibilidade de licitação, autorizado em 23/10/2023, por intermédio da IP ATI/DEGAT nº 11/2023, de 11/10/2023, conforme previsão orçamentária sob rubrica nº 3101700021, centro de custo nº BN000040000 – CCTI1, observado o disposto na Lei nº 13.303/2016 e no Regulamento Licitações e Contratos do Sistema **BNDES**, têm, entre si, justo e contratado o que se contém nas Cláusulas seguintes:

CLÁUSULA PRIMEIRA – OBJETO
O presente Contrato tem por objeto a prestação de serviços de suporte técnico com direito de atualização de versão de *software* para o Sistema Gerenciador de Banco de Dados (SGBD) composto pelos programas *Oracle Database Enterprise Edition*, *Oracle Diagnostics Pack*, *Oracle Partitioning*, *Oracle Real Application Cluster* e *Oracle Tuning Pack* na modalidade *Premier Support*, incluindo os serviços do *Oracle Priority Support*, conforme especificações constantes do Termo de Referência e da proposta apresentada pelo **CONTRATADO**, respectivamente, Anexos **I** e **II** deste Contrato.

Parágrafo Primeiro
Em caso de divergência entre os termos deste Contrato e do Termo de Referência, as disposições do Contrato prevalecerão sobre as disposições do Termo de Referência.

Classificação: Documento Ostensivo (Lei nº 13.303/2016)
Unidade Gestora: AJ1/JULIC/GLIC3

CONTRATO OCS Nº 159/2021
CONTRATO SAP Nº 4400004702

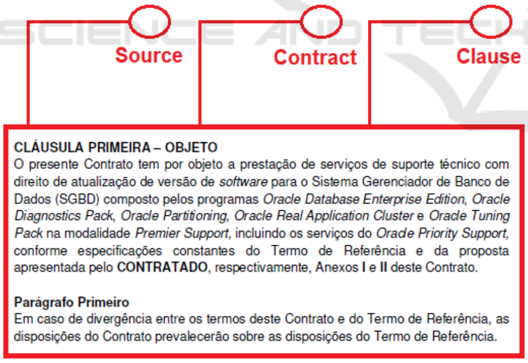
CONTRATO DE PRESTAÇÃO DE SERVIÇOS QUE ENTRE SI CELEBRAM O BANCO NACIONAL DE DESENVOLVIMENTO ECONÔMICO E SOCIAL – BNDES E IBM BRASIL INDÚSTRIA MÁQUINAS E SERVIÇOS LTDA, NA FORMA ABAIXO:

O **BANCO NACIONAL DE DESENVOLVIMENTO ECONÔMICO E SOCIAL – BNDES**, empresa pública federal, com sede em Brasília - DF e serviços no Rio de Janeiro – RJ, na Av. República do Chile, nº 100, CEP nº 20.031-917, inscrito no Cadastro Nacional de Pessoas Jurídicas sob o nº 33.657.248/0001-89, doravante denominado simplesmente **BNDES**, neste ato representado na forma do seu Estatuto Social; e **IBM BRASIL INDÚSTRIA MÁQUINAS E SERVIÇOS LTDA**, com sede na Avenida República do Chile 330, 11º e 12º andares, bloco 1 salas 1101 e 1201, e bloco 2 salas 1101 e 1201 – Centro – Rio de Janeiro -RJ, inscrita no Cadastro Nacional de Pessoas Jurídicas sob o nº 33.372.251/0001-56, doravante denominado(a) simplesmente **CONTRATADO**, neste ato representado na forma de seus atos constitutivos, em conformidade com o Procedimento de Inexigibilidade de Licitação nº 022/2021 - **BNDES**, com fundamento no artigo 30 da Lei nº 13.303/20216, autorizado em 03/08/2021, por intermédio da IP ATI/DESIS4 nº 06/2021, de 30/07/2021, publicado no DOU em 04/08/2021, seção 3, página 44, conforme previsão orçamentária sob rubrica nº 3101700040, centro de custo nº BN00004000, observado o disposto na Lei nº 13.303/2016 e no Regulamento Licitações e Contratos do Sistema **BNDES**, têm, entre si, justo e contratado o que se contém nas Cláusulas seguintes:

CLÁUSULA PRIMEIRA – OBJETO
O presente Contrato tem por objeto a prestação continuada de serviço de suporte técnico especializado, denominado AVP (*Accelerated Value Program Software*), para as ferramentas *IBM Content Manager*, *IBM Content Manager OnDemand*, *IBM Enterprise Records* (Grupo 1) e *IBM MQ Series* (Grupo 2) em uso no BNDES, conforme condições constantes das Especificações Técnicas e da Proposta apresentada pela Contratada, respectivamente, Anexos I e II deste Contrato.

CLÁUSULA SEGUNDA – VIGÊNCIA
O presente Contrato terá duração de 24 (vinte e quatro) meses, a contar da data de sua assinatura, podendo ser prorrogado, mediante aditivo, por períodos sucessivos, até o limite total de 5 (cinco) anos.

Figure 3: Chunking applied to Contracts.



CLÁUSULA PRIMEIRA – OBJETO
O presente Contrato tem por objeto a prestação de serviços de suporte técnico com direito de atualização de versão de *software* para o Sistema Gerenciador de Banco de Dados (SGBD) composto pelos programas *Oracle Database Enterprise Edition*, *Oracle Diagnostics Pack*, *Oracle Partitioning*, *Oracle Real Application Cluster* e *Oracle Tuning Pack* na modalidade *Premier Support*, incluindo os serviços do *Oracle Priority Support*, conforme especificações constantes do Termo de Referência e da proposta apresentada pelo **CONTRATADO**, respectivamente, Anexos **I** e **II** deste Contrato.

Parágrafo Primeiro
Em caso de divergência entre os termos deste Contrato e do Termo de Referência, as disposições do Contrato prevalecerão sobre as disposições do Termo de Referência.

Figure 4: Contracts metadata.

The LangChain SQL Agent (Langchain, 2024) has proven to be a highly flexible tool for interacting with the CMS database. Upon system startup, our SQL agent establishes an authenticated connection to the database and retrieves the schema. When it receives a user question, it performs Entity Recognition, maps those entities to the database tables and columns, and prepares the SQL statement.

Ensuring the safety of our SQL agent is central. We validate each generated query to ensure it does not contain harmful commands, such as 'UPDATES,' 'DROP TABLE,' 'INSERT,' or any other command that can alter the database, providing a sense of security about the system's integrity.

Finally, the output generated from the executed SQL statement goes to a prompt generation stage for further analysis of the LLM.

3.3 Applying Prompt Engineering

The *prompt* engineering technique provides a pattern for the style of responses and the reuse of the solution when accessing the LLM, as it provides instructions and context. Based on these observations, instructions were developed in the application to improve the responses. The instructions include basic guidelines, such as *"Do not use prior knowledge"*, which ensures that the responses are based only on *vectorstore* contracts, and specific instructions, such as *"Whenever you answer a question about a contract, provide the OCS number."* Thus, the question *"Do we have an Oracle Support contract?"* would have as a possible answer *"Yes, we have an Oracle Database Support contract. The OCS number is 278/2023."*.

Maintenance and guidelines on how to use the chat context were also applied to ensure uniformity and coherence. For example, we inform the expected style for responses: *"You should use a formal and objective tone."*, determining the style of LLM responses. Another guideline instructs LLM: *"Given the chat history and the question asked, construct the response completely, without the user needing to review the history"*.

Finally, the context passed to the LLM can be useful to establish the style of the answers according to the role of the user of the Q&A system. In the case of Contract*360*, we have three roles: 1) contract manager; 2) contract management support; and 3) manager of the contract management support unit. For each of these roles a specific context was defined, for example for role 3 we have: *"You are an assistant specialized in answering questions about administrative contracts, who provides management and summarized information about the contracts."*

With these three techniques we obtained more relevant answers. In the following section, we detail the implementation and the components used in the development of the system.

3.4 Applying Agents

In Contrato360, Agents play a pivotal role in orchestrating the flow of execution and enhancing the overall efficiency of the question-and-answer process. Also, considering the workflow on understanting the user query, an agent approach is a clever choice to implement this several specialized activities that needs to be taken in building the correct answer for the user. We designed three agents to implement this workflow.

As shown in figure 2, the Router Agent is central to its architecture, acting as the primary decision-making entity that orchestrates the flow of tasks needed to answer a user's question. The "Router Agent" decides if the user's question is related to the Contract Manager domain, e.g., "How are you?", "Will Bologna FC win the 2025 Champions League?" or "Who is the contract manager for the Database support?". An out-of-topic question is redirected to the LLM with a context limiting its role to the domain of contract management. In A question on the contract domain will follow our workflow to find a relevant answer.

In the sequel, the Router Agent sends the user question to two specialized agents: a) SQL agent and b) RAG agent. The RAG agent retrieves from the *vectorstore* chunks of documents similar to the user question. In parallel, a SQL agent retrieves form the CMS database content related to the user question. This architectural choice proved to be robust in the reports of the contract managers, as it semantically enriches the contract information, as shown in Figure 1.

One of the specialized agents in Contrato360 is the RAG (Retrieval Augmented Generation) Agent, responsible for retrieving relevant information from the contracts *vectorstore*. When directed by the Router Agent, the RAG Agent searches for similar data chunks that can help contextualize the question. Another specialized component is the SQL Agent, which handles queries requiring structured data extraction from the contracts database. Upon receiving routing instructions from the Router Agent, the SQL Agent executes SQL queries to retrieve specific data points relevant to the user's question.

With all textual and information retrieve, another "Router Agent" craft an answer. If needed to add an visual information, the Graph Agent and LLM Answer Generation Agent add further depth to Contrato360's response capabilities. The Graph Agent is tasked with creating visual representations, such as charts, when the Router Agent determines that a visual answer would better serve the user's needs. This agent ensures that complex data can be conveyed in a clear and understandable format, enhancing user comprehension. Meanwhile, the LLM Answer Generation Agent works closely with the prompt generation module to produce coherent and contextually relevant textual responses. Together, these agents provide a multi-faceted approach to answering questions, combining data retrieval, visualization, and language generation to deliver comprehensive solutions.

4 ARCHITECTURE

The architecture of the Contrato360 application illustrates a comprehensive system designed to facilitate a question-answering application that integrates Large Language Models (LLMs), document processing, and databases. The architecture consists of three main layers: the User Interface Layer, the Backend Layer, and the Language Model Integration Layer, each playing its role in delivering accurate and context-aware responses to users.

The User Interface Layer is represented by the User Interface (Streamlit), which serves as the frontend of the application. This layer provides an interactive platform where users can input their queries and view the responses generated by the system. The interface directly communicates with the backend layer, sending user inputs for processing and displaying the responses generated by the various integrated components.

At the heart of the system lies the Backend Layer, which is primarily managed by the Backend Agents

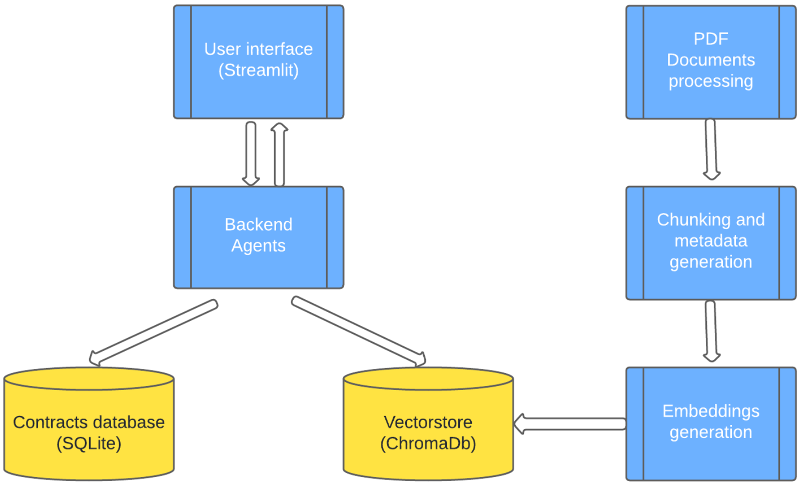

User interface (Streamlit)
PDF Documents processing
Backend Agents
Chunking and metadata generation
Contracts database (SQLite)
Vectorstore (ChromaDb)
Embeddings generation

Figure 5: Application architecture.

(Python and Langchain). This layer orchestrates interactions between the document processing, vector storage, contracts database, and the language model integration layer. The backend layer leverages Python and Langchain to handle the logic, task execution, and chat functionalities, particularly through OpenAI's chat models. It processes user inputs received from the interface and interacts with both the Contracts Database and Vectorstore (ChromaDb) to retrieve relevant information necessary for formulating comprehensive answers.

Within the backend layer, the Contracts Database (SQLite) serves as the structured data source, storing structured information related to contracts. This component allows the system to handle contract-related questions by processing SQL queries generated by the backend agents. The contracts database responds to these queries with relevant data, which is then used to construct natural language responses for the user.

The Vectorstore (ChromaDb) is another vital component of the backend layer, acting as a storage solution for vectorized data, including document embeddings. It plays a key role in efficient similarity searches and retrieval tasks, enhancing the system's ability to provide context-aware responses. The backend agents utilize the Vectorstore to match user queries against stored embeddings, enabling advanced semantic search capabilities. This component also stores embeddings generated from document processing, ensuring that data is readily available for future query matching.

The Language Model Integration Layer is responsible for transforming and embedding data for use within the system. This layer includes the PDF Documents Processing module, which ingests and preprocesses documents, particularly PDFs, to make them suitable for use within the application. This step involves reading and extracting text and relevant metadata, preparing the content for the next stages of processing. The Chunking and Metadata Generation component further refines the documents by dividing them into manageable chunks and generating metadata that improves retrieval efficiency, ensuring that the data is optimally split for better embedding generation and response times.

The final stage of the language model integration layer is the Embeddings Generation module, which converts the chunked documents and metadata into vector embeddings using LLM-based models like OpenAI Embeddings. These embeddings capture the semantic nuances of the text, facilitating efficient search and retrieval tasks within the system. Once generated, these embeddings are stored in the Vectorstore (ChromaDb), where they can be accessed for matching against user queries.

The overall workflow begins when a user inputs a question through the User Interface Layer, initiating a sequence of processes across the backend and language model integration layers. The backend agents handle query processing, interacting with the Contracts Database for SQL queries and performing semantic searches using embeddings from the Vectorstore. The document processing involves preprocessing PDFs, chunking the content, and generating embeddings that are then stored for efficient retrieval. The backend agents combine data retrieved from the contracts database and the Vectorstore to generate a coherent response, which is then presented back to the user through the User Interface Layer.

This architecture effectively combines the User Interface Layer, Backend Layer, and Language Model Integration Layer, enabling Contrato360 to function as a robust and powerful application for answering questions based on complex data sources. The seamless integration of multiple technologies ensures that users receive accurate and contextually relevant responses, enhancing the overall functionality and usability of the system.

5 EVALUATION

The experiment to validate the application was conducted by two IT contract specialists from BNDES. The system was prepared with 75 contracts (PDFs and data from the contract system). And to validate the relevance of the answers, *benchmark* questions were prepared, from two distinct groups: ”direct” and ”indirect” questions. ”Direct” questions are those that can be answered through the PDFs and their metadata. ”Indirect” questions are those that obtain better relevance when searched in the contract system data. In Tables 1 and 2 we present the users’ perception of the quality of the answers. In the evaluation, the relevance of the answers was categorized as ”Correct” and ”Incomplete”.[1]

We can observe that for the ”direct” questions the system presents relevant answers for all experiments. However, in the ”indirect” questions, despite being satisfactory, the results in one specific question were limited and incomplete. In our evaluation, these questions require a more elaborate semantic evaluation. In the first case, we realized that the concept of ”Waiver of Bidding” was not well captured. We believe that an adjustment in the queries and/or in the prompt can add this type of semantics.

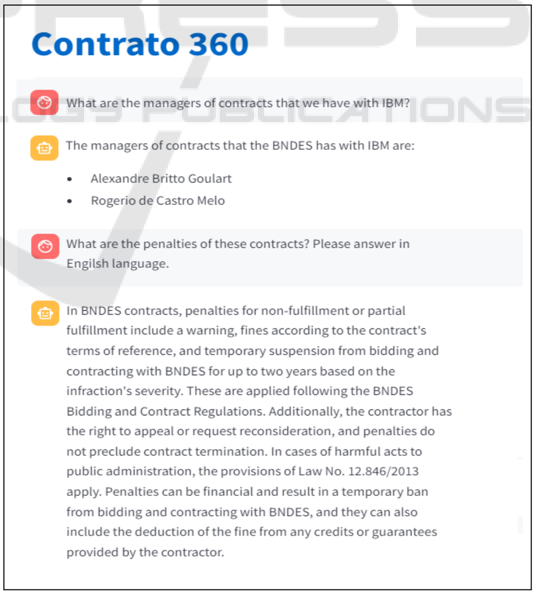
Table 1: Direct Questions.

Question	Correct	Incomplete
What is the subject of the OCS nnn/yy contract?	10	-
Do we have any contract whose subject is xxxx?	9	1
Do we have any contract with the supplier xxx?	10	-
Who is the manager of the OCS nnn/yy contract?	10	-
Who is the supplier of the nnn/yy contract?	10	-
What is the term of the OCS nnn/yy contract?	10	-

Table 2: Indirect Questions.

Question	Correct	Incomplete
How many active IT contracts do we currently have?	10	-
List the contracts that will end in the year yy?	10	-
How many contracts do we have with supplier xxxx?	10	-
How many contracts have we signed due to inflexibility?	9	1
How many DLs (Exemptions from Tenders) were contracted in yy?	-	10
Who are the managers of the contracts we have with company xxxx?	8	2
How many contracts does employee xxxx have under his/her management?	8	2
Show a summary of contract nnn/yy.	10	-

A key aspect observed from the users is the solution’s capability to combine answers from both the structured data store and the contract’s texts. This integration is perceived as a significant time-saving feature, as users typically need to locate the relevant contracts, open the respective PDFs, and manually search for additional information. The example below illustrates this. It identifies contract managers and outlines the penalties associated with contractual noncompliance. The system’s ability to deliver precise, context-relevant answers from contracts highlights its effectiveness in reducing manual search efforts for users.



Contrato 360

What are the managers of contracts that we have with IBM?

The managers of contracts that the BNDES has with IBM are:

- Alexandre Britto Goulart
- Rogerio de Castro Melo

What are the penalties of these contracts? Please answer in Engilsh language.

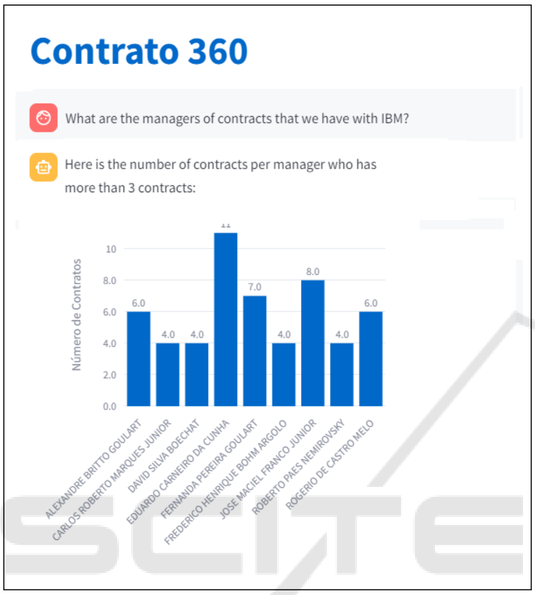
In BNDES contracts, penalties for non-fulfillment or partial fulfillment include a warning, fines according to the contract's terms of reference, and temporary suspension from bidding and contracting with BNDES for up to two years based on the infraction's severity. These are applied following the BNDES Bidding and Contract Regulations. Additionally, the contractor has the right to appeal or request reconsideration, and penalties do not preclude contract termination. In cases of harmful acts to public administration, the provisions of Law No. 12.846/2013 apply. Penalties can be financial and result in a temporary ban from bidding and contracting with BNDES, and they can also include the deduction of the fine from any credits or guarantees provided by the contractor.

Figure 6: Contracts Q&A Streamlit application.

In fact, by directly addressing questions with specific details, the system saves time and improves the user experience, as users can quickly access critical information without sifting through lengthy documents. Finally, the system’s ability to automatically generate graphs using its Plotly agent, when a table of

[1] A third category would be ”Incorrect”, but this option was not obtained in any of the questions.

values is included in the response, has been positively received by users. This feature not only provides immediate visual insights, enhancing the understanding of the data, but also supports users in creating professional presentations. The integration of dynamic graph generation into the query response process significantly enriches the user experience, allowing for a more comprehensive analysis and efficient communication of contract-related information.



Contrato 360

What are the managers of contracts that we have with IBM?

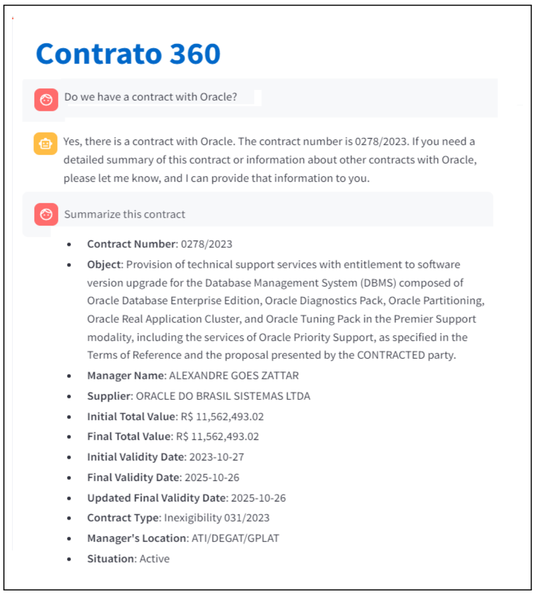
Here is the number of contracts per manager who has more than 3 contracts:

Figure 7: Plotly Agent.

6 CONCLUSIONS

We developed a Q&A application in the domain of service and product contracts, using PDF contracts and data from the Contract Management System as information sources. In this development, we employed four techniques to improve the relevance of the answers: 1) Augmented Retrieval (RAG) combined with semantic augmentation using metadata to retrieve information from PDFs; 2) Text-to-SQL, aggregating dynamic information from the contracts made available in the Contract Management System; 3) Prompt Engineering to contextualize, instruct and direct the answers produced by the LLM; and 4) Agents to call the most appropriate approach depending on query context and determining the flow of execution of tasks in the system.

The 8 demonstrates the ability of Contrato360 in retrieving and summarizing contract information related to Oracle through a question-and-answer interface. When asked if there is a contract with Oracle, the system efficiently identifies the relevant contract, numbered 0278/2023, and provides a concise summary of its key details stored in the database. The summarized information includes the contract's object, which covers technical support and software upgrades for Oracle's Database Management System (DBMS), details about the contract manager, supplier, total value, validity dates, and the current situation. This functionality highlights the system's ability to streamline access to specific contract data, facilitating quick and accurate information retrieval for users by directly interacting with the database through natural language queries



Contrato 360

Do we have a contract with Oracle?

Yes, there is a contract with Oracle. The contract number is 0278/2023. If you need a detailed summary of this contract or information about other contracts with Oracle, please let me know, and I can provide that information to you.

Summarize this contract

- **Contract Number**: 0278/2023
- **Object**: Provision of technical support services with entitlement to software version upgrade for the Database Management System (DBMS) composed of Oracle Database Enterprise Edition, Oracle Diagnostics Pack, Oracle Partitioning, Oracle Real Application Cluster, and Oracle Tuning Pack in the Premier Support modality, including the services of Oracle Priority Support, as specified in the Terms of Reference and the proposal presented by the CONTRACTED party.
- **Manager Name**: ALEXANDRE GOES ZATTAR
- **Supplier**: ORACLE DO BRASIL SISTEMAS LTDA
- **Initial Total Value**: R$ 11,562,493.02
- **Final Total Value**: R$ 11,562,493.02
- **Initial Validity Date**: 2023-10-27
- **Final Validity Date**: 2025-10-26
- **Updated Final Validity Date**: 2025-10-26
- **Contract Type**: Inexigibility 031/2023
- **Manager's Location**: ATI/DEGAT/GPLAT
- **Situation**: Active

Figure 8: Contract Summarization.

In our experiment, we addressed an initial set of questions that were able to produce a robust system that meets current user needs. However, exploring other questions in depth will allow us to enrich the metadata and the set of queries that extract information from traditional systems.

Finally, to consolidate the techniques developed to address our application, we envision that building a system in a different problem domain may shed light on limitations and the possible need for refinement or adaptation. Such future exploration will not only reinforce confidence in the implementation of these techniques in real-world scenarios, but also pave the way for their optimization and possible customization for specific domains, ultimately increasing the utility and impact of LLMs in enterprise applications.

REFERENCES

Chen, J., Lin, H., Han, X., and Sun, L. (2024). Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17754–17762.

Fan, W., Ding, Y., Ning, L., Wang, S., Li, H., Yin, D., Chua, T.-S., and Li, Q. (2024). A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6491–6501.

Feng, Z., Feng, X., Zhao, D., Yang, M., and Qin, B. (2024). Retrieval-generation synergy augmented large language models. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 11661–11665. IEEE.

Gao, D., Wang, H., Li, Y., Sun, X., Qian, Y., Ding, B., and Zhou, J. (2023a). Text-to-sql empowered by large language models: A benchmark evaluation. *arXiv preprint arXiv:2308.15363*.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., and Wang, H. (2023b). Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Giray, L. (2023). Prompt engineering with chatgpt: a guide for academic writers. *Annals of biomedical engineering*, 51(12):2629–2633.

Jeong, C. (2023). A study on the implementation of generative ai services using an enterprise data-based llm application architecture. *arXiv preprint arXiv:2309.01105*.

Jin, H., Huang, L., Cai, H., Yan, J., Li, B., and Chen, H. (2024). From llms to llm-based agents for software engineering: A survey of current, challenges and future. *arXiv preprint arXiv:2408.02479*.

Langchain (2024). Langchain agents documentation. https://python.langchain.com/v0.1/docs/use_cases/sql/agents/. Accessed: 2024-09-06.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Li, H., Su, Y., Cai, D., Wang, Y., and Liu, L. (2022). A survey on retrieval-augmented text generation. *arXiv preprint arXiv:2202.01110*.

Liu, A., Hu, X., Wen, L., and Yu, P. S. (2023). A comprehensive evaluation of chatgpt's zero-shot text-to-sql capability. *arXiv preprint arXiv:2303.13547*.

Mialon, G., Dessì, R., Lomeli, M., Nalmpantis, C., Pasunuru, R., Raileanu, R., Rozière, B., Schick, T., Dwivedi-Yu, J., Celikyilmaz, A., et al. (2023). Augmented language models: a survey. *arXiv preprint arXiv:2302.07842*.

OpenAI (2023a). Chatgpt fine-tune description. https://help.openai.com/en/articles/6783457-what-is-chatgpt. Accessed: 2024-03-01.

OpenAI (2023b). Chatgpt prompt engineering. https://platform.openai.com/docs/guides/prompt-engineering. Accessed: 2024-04-01.

Pinheiro, J., Victorio, W., Nascimento, E., Seabra, A., Izquierdo, Y., Garcıa, G., Coelho, G., Lemos, M., Leme, L. A. P. P., Furtado, A., et al. (2023). On the construction of database interfaces based on large language models. In *Proceedings of the 19th International Conference on Web Information Systems and Technologies - Volume 1: WEBIST*, pages 373–380. INSTICC, SciTePress.

Saeed, M., De Cao, N., and Papotti, P. (2023). Querying large language models with sql. *arXiv preprint arXiv:2304.00472*.

Seabra, A., Nepomuceno, J., Lago, L., Ruberg, N., and Lifschitz, S. (2024). Contrato360: uma aplicação de perguntas e respostas usando modelos de linguagem, documentos e bancos de dados. In *Anais do XXXIX Simpósio Brasileiro de Bancos de Dados*.

Singh, A., Ehtesham, A., Kumar, S., and Khoei, T. T. (2024). Enhancing ai systems with agentic workflows patterns in large language model. In *2024 IEEE World AI IoT Congress (AIIoT)*, pages 527–532. IEEE.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, M., Wang, M., Xu, X., Yang, L., Cai, D., and Yin, M. (2023). Unleashing chatgpt's power: A case study on optimizing information retrieval in flipped classrooms via prompt engineering. *IEEE Transactions on Learning Technologies*.

Wang, Z., Wang, Z., Le, L., Zheng, H. S., Mishra, S., Perot, V., Zhang, Y., Mattapalli, A., Taly, A., Shang, J., et al. (2024). Speculative rag: Enhancing retrieval augmented generation through drafting. *arXiv preprint arXiv:2407.08223*.

White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., and Schmidt, D. C. (2023). A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. (2023). React: Synergizing reasoning and actin in langage models. *arXiv preprint arXiv:2210.03629v3*.