Web Application for Retrieval-Augmented Generation: Implementation and Testing

Irina Radeva *, Ivan Popchev *, Lyubka Doukovska, Miroslava Dimitrova

Posted Date: 15 March 2024

doi: 10.20944/preprints202403.0844.v1

Keywords: Retrieval-augmented generation (RAG); open-source large language models (LLMs); 30 Antelope blockchain; IPFS; Ollama; LangChain; smart agriculture

Article

Web Application for Retrieval-Augmented Generation: Implementation and Testing

Irina Radeva [1,*], Ivan Popchev [2], Lyubka Doukovska [3] and Miroslava Dimitrova [4]

[1] Intelligent systems department, Institute of Information and Communication Technologies – Bulgarian Academy of Sciences; 1113 Sofia, Bulgaria; irina.radeva@iict.bas.bg
[2] Bulgarian Academy of Sciences; 1040 Sofia, Bulgaria; ivan.popchev@iict.bas.bg
[3] Intelligent systems department, Institute of Information and Communication Technologies – Bulgarian Academy of Sciences; 1113 Sofia, Bulgaria; lyubka.doukovska@iict.bas.bg
[4] Intelligent systems department, Institute of Information and Communication Technologies – Bulgarian Academy of Sciences; 1113 Sofia, Bulgaria; miroslava.dimitrova@iict.bas.bg
* Correspondence: irina.radeva@iict.bas.bg

Abstract: The purpose of this paper is to explore the implementation of Retrieval-Augmented Generation (RAG) technology with open-source Large Language Models (LLMs). For this purpose, a web-based application named PaSSER that integrates RAG with three models - Mistral:7b, Llama2:7b: Orca2:7b, is developed. In its development various technologies are used, including Blockchain for managing and storing assessment results of built-in testing modules. PaSSER employs a set of evaluation metrics for LLMs' performance - METEOR, ROUGE, BLEU, Perplexity, Cosine similarity, Pearson correlation, and F1 score. The specialized knowledge base is in the smart agriculture domain. As an illustration, the paper presents results and analysis of two tests. First one is assessing the performance of LLMs across different hardware configurations, the other is determining the model that delivers most accurate and contextually relevant responses within the RAG. The paper also discusses the integration of blockchain technology with LLMs to manage and store assessment results within a blockchain environment. The tests revealed that Mistral:7b demonstrates the best results across most of the evaluated metrics. The discussion section comments on technical and hardware considerations that affect the performance of LLMs. Future development will focus in leveraging new larger pre-trained open-source LLMs, exploring different finetuning approaches, and further integration with Blockchain and IPFS.

Keywords: retrieval-augmented generation (RAG); open-source large language models (LLMs); Antelope blockchain; IPFS; Ollama; LangChain; smart agriculture

1. Introduction

In response to the increasing volume of digitally generated and stored information, the field of document retrieval has undergone significant changes over the years. Initially, document retrieval systems relied mainly on Boolean logic and exact matching algorithms [1]. However, these early systems were not flexible enough and failed to capture the goals of user requests in detail. This led to the development of vector space models [2] and the inverse document frequency (TF-IDF) algorithms. These models improved retrieval performance since they took into account the frequency and distribution of terms within documents and across a corpus. However, they fail to understand the context and semantic relationships between words, which results in mismatches between user queries and retrieved documents.

The search for more advanced solutions led to the development of Latent Semantic Indexing (LSI) and Latent Dirichlet Allocation (LDA) [3]. They aim to capture the main semantic structures of the text. By analysing word distribution patterns in documents, these techniques seek to improve the relevance of retrieved documents to user queries. Despite their progress, these models require massive computational resources and are difficult to adapt to the dynamic and evolving landscape of human cognition.

 doi:10.20944/preprints202403.0844.v1

The development of Retrieval-Augmented Generation (RAG) represents a significant advancement in the field of natural language processing (NLP). It combines the strengths of parametric and non-parametric memory to improve language models. The main concepts of RAG [4] involve a combination of pre-trained language models with external knowledge retrieval, enabling dynamic, informed content generation. Applications cover various NLP tasks, demonstrating improvements in question answering, fact verification, and more. This is achieved by enabling models to access and leverage vast knowledge repositories effectively. Its foundational concept was introduced to integrate external data retrieval into the generative processes of LLMs, by enhancing their capability to generate accurate and contextually relevant responses. This process involves querying an external data source to obtain relevant information before generating text. This ensures that responses are grounded in reliable evidence and thus improving significantly the models' output quality.

In this paper, RAG is viewed as a *technology*, rather than a mere *method*. This distinction is due to the paper's emphasis on the applied, practical, and integrative aspects of RAG in the field of NLP.

The advanced of RAG technology has unfolded across several phases, which overlapped with advances in the Transformer architecture. Initially focused on augmenting language models with additional knowledge during pre-training, RAG's development entered a phase of rapid growth, where RAG approaches began combining hybrid models that include retrieval and generation techniques to improve performance and adaptation across various NLP tasks.

RAG establishes a significant development in addressing limitations intrinsic to LLMs, such as hallucinations, outdated knowledge, and untraceable reasoning processes. RAG has evolved through various phases, far beyond its initial concept to enrich generative tasks by leveraging external databases, thus significantly impacting the field of NLP.

Nevertheless, challenges such as integrating diverse knowledge sources, ensuring the relevance and accuracy of retrieved information, and efficiently training and fine-tuning these hybrid models, still persist. As RAG continues to evolve, addressing these challenges remains critical for improving the capabilities and applications of language models in complex, knowledge-driven tasks [5].

The integration of RAG into LLMs has eased their application in more practical settings, particularly for chatbots and other AI-driven interactive systems. By addressing the issue of generating incorrect or irrelevant information, RAG has made LLMs more reliable and effective for real-world applications, particularly those requiring up-to-date information or domain-specific knowledge.

The purpose of this paper is to explore the implementation of Retriever-Augmented Generation (RAG) technology with open source Large Language Models (LLMs). In order to support this research, a web-based application PaSSER that allows the integration, testing and evaluation of such models in a structured environment has been developed.

The paper discusses the architecture of the web application, the technological tools used, the models selected for integration and the set of functionalities developed to operate and evaluate these models. The evaluation of the models has two aspects: operation on different computational infrastructures and performance in text generation and summarization tasks.

The domain of smart agriculture is chosen as the empirical domain for testing the models. Furthermore, the web application is open source, which promotes transparency and collaborative improvement. Detailed guide on installing and configuring the application, the datasets generated for testing purposes and the results of the experimental evaluations are provided and available at GitHub [6].

The application allows adaptive testing of different scenarios. It integrates three of the leading LLMs, Mistral:7b, Llama2:7b and Orca2:7b, which do not require significant computational resources. A set of standard NLP metrics - METEOR, ROUGE, BLEU, Laplace and Lidstone's Perplexity, Cosine similarity, Pearson correlation coefficient, and F1 score - is selected for a thorough evaluation of the models' performance.

The paper contributes to the field of RAG research in several areas:

 doi:10.20944/preprints202403.0844.v1

1. By implementing the PaSSER application, the study provides a practical framework that can be used and expanded upon in future RAG research.

2. The paper illustrates the integration of RAG technology with blockchain, enhancing data security and verifiability, which could inspire further exploration into the secure and transparent application of RAG systems.

3. By comparing different LLMs within the same RAG framework, the paper provides insights into the relative strengths and capabilities of the models, contributing the knowledge on model selection in RAG contexts.

4. The focus on applying and testing within the domain of smart agriculture adds to the understanding of how RAG technology can be tailored and utilized in specific fields, expanding the scope of its application and relevance.

5. The use of open-source technologies in PaSSER development, allows the users to review and trust the application's underlying mechanisms. More so, it enables collaboration, flexibility to adapt for specific needs or research goals, reduces development costs, facilitates scientific accuracy by enabling exact replication of research setups, and serves as a resource for learning about RAG technology and LLMs in practical scenarios.

Further the paper is organised as follows: *Section 2* provides an overview of the development, implementation, and functionalities of the PaSSER Web App; *Section 3* discusses selected standard NLP metrics used to measure RAG performance; *Section 4* presents the results of tests performed on the three language models using two test features of the PaSSER App; In the *discussion section 5*, the limitations and influencing factors highlighted during the testing of the language models in the context of the PaSSER application are discussed; The *conclusion section 6* summarises the results, the potential of the web application as a tool for evaluating and exploring RAG technology, the main contributions and future directions for development.

2. Web Application Development and Implementation

This section describes the development, implementation and features of the PaSSER Web App, which henceforward will be referred to as PaSSER App.

There are different ways and techniques to work with LLMs and the RAG technology (Python scripts, desktop or web apps), but from a user's perspective, it is most convenient to use the web interface as it is independent of software platforms, where access to external resources is implemented through APIs.

Figure 1 depicts the operational framework for the Web App, outlining its core components which include both local and remote deployments of a server and the web application itself. It incorporates blockchain and IPFS networks, integral to the Smart Crop Production data exchange (SCPDx) platform, described in detail in [7], and [8]. The development of the SCPDx platform was preceded by a number of studies related to the choice of blockchain platform [9], blockchain-enabled supply-chain modelling for a smart crop production framework [10], blockchain oracles integration[11]. PaSSER App is a complementary project to the SCPDx platform. The platform is based on private Antelope blockchains/IPFS networks. It aims to support the integration of the exchange of information and data obtained or generated as a result of the use of different technologies in smart agriculture.

The PaSSER App communicates with Ollama, Chroma DB, Python scripts, Antelope blockchain, and IPFS through their respective APIs. Currently, the IPFS network is not utilized within the framework; however, future enhancements aim to incorporate it to enable storage and content distribution, which will support the fine-tuning processes of LLMs.

 doi:10.20944/preprints202403.0844.v1

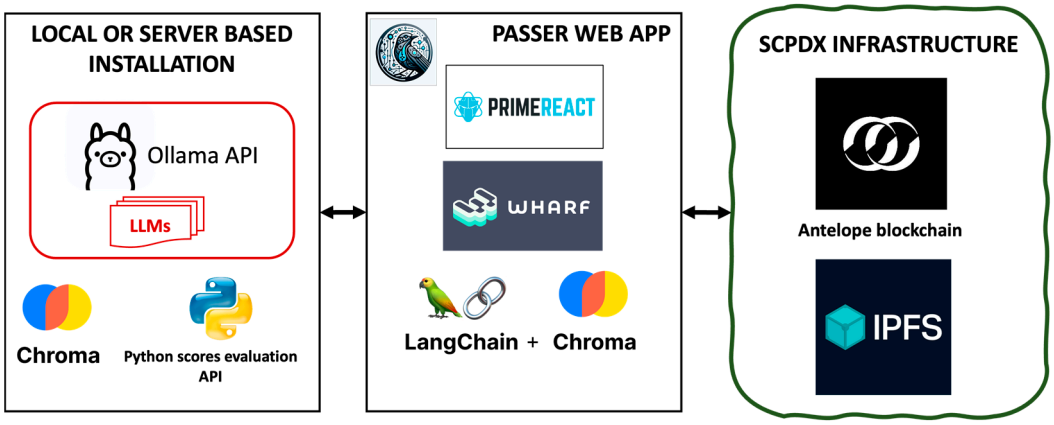


Figure 1. PaSSER App framework.

The PaSSER App is developed in JavaScript, leveraging the Primereact library for its user interface components. To facilitate interaction with the Antelope blockchain network, it employs the WharfKit library. The LangChain library is integrated for engagement with RAG and LLMs. The application is hosted on an Apache web server, enabling communication between the PaSSER App, the LLMs, and the SCPDx infrastructure.

The server component of the PaSSER App includes a vector database, LLM API, and a score evaluation API. The deployment uses the Ollama API to interface with various LLMs, supporting diverse operating systems including UBUNTU, Windows, and macOS. This API grants users the flexibility to manage and interact with different open-source LLMs. Specifically, Ollama is deployed on an Ubuntu server equipped with 128GB RAM, not utilizing a GPU, and also configured locally on a Mac M1 with 16GB RAM and a 10-core GPU.

ChromaDB [12] is used to work with vector databases. ChromaDB is an open-source vector database for applications that utilize large language models (LLMs). It supports multiple programming languages such as Python SDK, JavaScript, Ruby, Java, Go, and others. The database is licensed under the Apache 2.0 license. ChromaDB's architecture is suited for applications that require semantic search capabilities. An embedding refers to the transformation of text, images, or audio into a vector of numbers, which represents the essence of the content in a way that can be processed and understood by machine learning models. In this implementation ChromaDB is installed on a macOS based server and locally on a Mac M1 16GB RAM 10 GPU.

Two Python scripts are developed: first one for computing various evaluation metrics using libraries such as NLTK [13], torch [14], numpy [15], rouge [16], transformers [17], and scipy [18], and the other for logging model performance data, using the Pyntelope library [19] for blockchain interactions. These scripts are operational on an Ubuntu server and are accessible via GitHub [6], facilitating the analysis and the recording of performance metrics within a blockchain framework to ensure data integrity and reproducibility of results.

2.1. LLMs Integration

In this implementation of PaSSER, the Mistral:7b, Llama2:7b, and Orca2:7b models are selected. These models have the same volume parameters and are suitable to be installed and used on hardware configurations with medium computational capacity, while giving good enough results. A brief description of the models is presented below.

Mistral:7b (https://mistral.ai/news/announcing-mistral-7b/) is a language model developed by Mistral AI with a capacity of 7.3 billion parameters. Available under the Apache 2.0 license, which means it can be used without restrictions. The model can be downloaded and used anywhere, including locally, and deployed on any cloud (AWS/GCP/Azure) using LLM inference server and skypilot.

It's structured to process language data, facilitating a wide range of text-based applications. This model incorporates specific attention mechanisms, namely Grouped-query Attention (GQA) and

 doi:10.20944/preprints202403.0844.v1

Sliding Window Attention (SWA), aimed at optimizing the processing speed and managing longer text inputs. These technological choices are intended to improve the model's performance while managing computational resources effectively [20–22].

Llama2:7b (https://llama.meta.com/llama2/) is a series of large language models developed by Meta, offering variations in size from 7 billion to 70 billion parameters. The 7 billion parameter version, Llama 2:7b, is part of this collection and is designed for a broad range of text-based tasks, from generative text models to more specific applications like chatbots. The architecture of Llama 2 models is auto-regressive, utilizing an optimized transformer structure. These models have been pretrained on a mix of publicly available online data and can be fine-tuned for specific applications. They employ an auto-regressive language model framework and have been optimized for various natural language processing tasks, including chat and dialogue scenarios through supervised fine-tuning (SFT) and reinforcement learning with human feedback (RLHF) [23].

Llama2:7b models have been trained on significant computational resources and have a considerable carbon footprint, which Meta has committed to offsetting as part of their sustainability program. The 7b model specifically consumed 184,320 GPU hours and emitted an estimated 31.22 tCO2eq during its training, with all emissions directly offset. These models are intended for both commercial and research purposes, and while they have been primarily tested in English, they carry the potential for a broad array of applications. Meta has provided a custom commercial license for Llama 2, and details on accessing this license can be found on their website.

Orca2:7b, Orca 2 [24], developed by Microsoft, is a finetuned version of the Llama 2 model with two versions: one with 7 billion and the other with 13 billion parameters. It focuses on improving the reasoning abilities of smaller language models through enhanced training methods. By employing high-quality synthetic data for training, Orca 2 is designed to master various reasoning techniques such as step-by-step processing and recall-then-generate strategies. This synthetic training data is crafted to guide the model in adopting different solution strategies appropriate for varied tasks, aiming to optimize the model's problem-solving approach.

The model is a result of research aimed at leveraging the advancements in large language models to boost the performance of smaller models, making them more efficient and versatile in handling tasks that require complex reasoning. The initiative behind Orca 2 is to provide a resource for research into the development, evaluation, and alignment of language models, particularly focusing on smaller models that can perform efficiently across a range of tasks.

Orca 2 is available for public use and research, underscoring Microsoft's commitment to advancing the field of AI and language model technology. It represents an effort to explore how smaller models can be enhanced to approach a variety of tasks more effectively, without the extensive computational and environmental costs associated with larger models.

2.2. PaSSER App Functionalities

The site map for the PaSSER App is depicted in Figure 2, detailing its core features. These include user authentication (Login), system configuration (Configuration Setup), creation and management of vector stores from various sources (text, PDF files, and websites), and functionalities for engaging with the stored data through Q&A chat and RAG Q&A chat based on the established vector stores. Additionally, a testing module is outlined to evaluate the application's functionalities and performance.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 March 2024 doi:10.20944/preprints202403.0844.v1

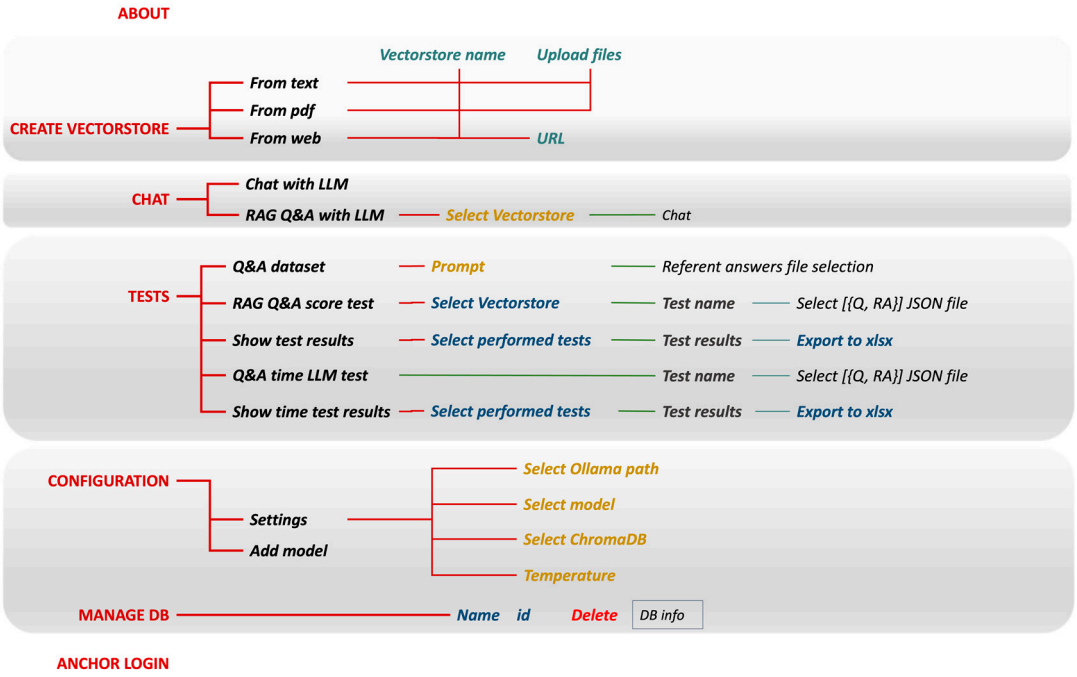


Figure 2. PaSSER site map. The figure provides a structured flowchart of the PaSSER App's user interface, mapping out the navigation pathways for various functionalities, such as the creation and management of a vector store, chat interactions, testing protocols, configuration settings, database management, and user authentication. Each section outlines the sequence of user actions and choices, from inputting data to exporting test results, configuring system settings, and maintaining the database.

The *Create vectorstore* menu, as depicted in Figure 3, outlines the process of converting raw textual data into a structured, query-able vector space using LangChain. This transformation NLP and vector embedding techniques to convert text into a format amenable to vector operations. Users can source textual data from text files, PDFs, and websites. The outlined procedure for vectorstore creation is standardized across these data types, ensuring consistency in processing and storage. At the current phase, automatic retrieval of information from websites (scrapping) is considered impractical due to the necessity for in-depth analysis of website structures, and the requirement for extensive manual intervention to adequately structure the retrieved text. This process involves understanding varied and complex web layouts, imposing a tailored approach to effectively extract and organize data.

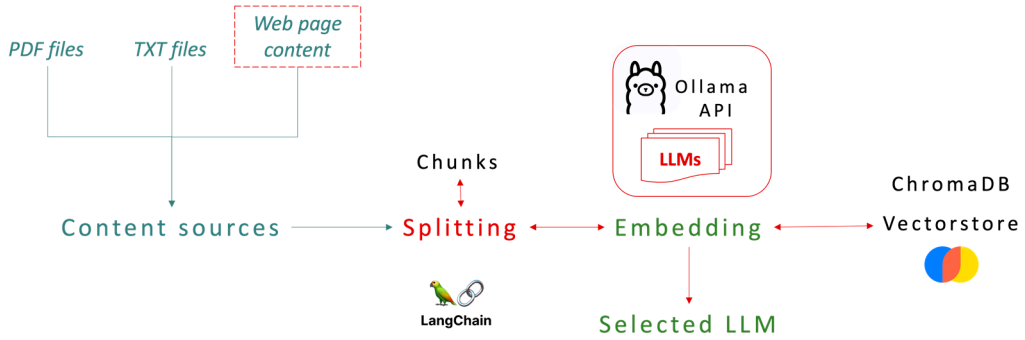


Figure 3. Vectorstore construction workflow.

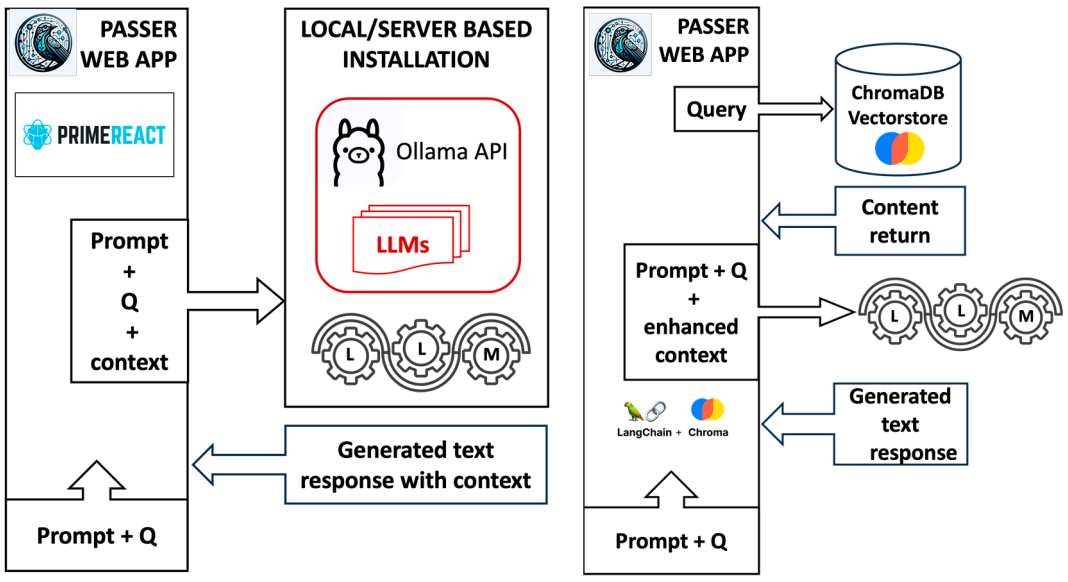
The process for creating a vectorstore involves the following steps, which are common to all three source types:

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 March 2024 doi:10.20944/preprints202403.0844.v1

1. *Cleaning and standardizing text data.* This is achieved by removing unnecessary characters (punctuation and special characters). Converting the text to a uniform size (usually lower case). Separating the text into individual words or tokens. In the implementation considered here, the text is divided into chunks with different overlap.
2. *Vector embedding.* The goal is to convert tokens (text tokens) into numeric vectors. This is achieved by using pre-trained word embedding models from selected LLMs (in this case Misrtal:7b, Llama2:7b, and Orca2:7b). These models map words or phrases to high-dimensional vectors. Each word or phrase in the text is transformed into a vector that represents its semantic meaning based on the context in which it appears.
3. *Aggregating embeddings* for larger text units to represent whole sentences or documents as vectors. It can be achieved by simple aggregation methods (averaging the vectors of all words in a sentence or document) or by using sentence transformers or document embedding techniques that take into account the more consistent and contextual nature of words. Here transformers are used which are taken from the selected LLMs.
4. *Create a vectorstore* to store the vector representations in a structured format. The data structures used are optimized for operations with high-dimensional vectors. ChromaDB is used for the vectorstore.

There are two distinct submenus in the Chat functionality. The first, '***Q&A Chat***' facilitates computational linguistics of LLMs to simulate a conversational exchange by interpreting and responding to user inquiries. For this interaction, the user navigates through the settings menu to select an LLM of choice via the Ollama API access point, thus activating the model for the chat session.

Figure 4 defines the PaSSER App's mechanisms for processing user queries and generating responses. Upon a user posting a question, the App formulates a query that interfaces with the Ollama API (as shown in Figure 4a). The corresponding LLM processes the query, formulates a response, and concurrently provides system performance data, including metrics such as total load and evaluation timeframes. Additionally, a numerical array captures the contextual backdrop of the query and the response, drawn from previous dialogue or related data, which the LLM utilizes similar to short-term memory to ensure response relevance and coherence. While the capacity of this memory is limited and not the focus of the current study, it is pivotal in refining responses based on specific contextual elements such as names and dates. The App enables saving this context for continued dialogue and offers features for initiating new conversations by purging the existing context.



(a) Q&A chat (b) RAG-based Q&A chat

 doi:10.20944/preprints202403.0844.v1

Figure 4. Chat types workflow. Figure 4 (a) represents a general Q&A chat workflow with direct input without the augmented context provided by a vectorstore. Figure 4 (b) illustrates the workflow of a Q&A chat using the RAG technique, which employs a pre-built vectorstore for data retrieval and response generation.

The ***'RAG Q&A Chat'*** feature depicted in Figure 4b facilitates interaction with an existing vectorstore. Users commence the chat by entering a query, triggering the LangChain library to fetch related data from the chosen vectorstore. This data informs the subsequent query to the LLM, integrating the original question, any prompts, and a context enriched by the vectorstore's information. The LLM then generates a response. Within the app, a dedicated memory buffer recalls history, which the LLM utilizes as transient context to ensure consistent and logical responses. The limited capacity of this memory buffer and its impact on response quality are acknowledged, though not extensively explored in this study. In the '***RAG Q&A Chat'***, context-specific details like names and dates are crucial for enhancing the relevance of responses.

The '***Tests***' feature is designed to streamline the testing of various LLMs within a specific knowledge domain. It involves following steps:

1. Selection of a specific knowledge base in a specific domain.

With '***Create vectorstore'***, the knowledge base is processed and saved in the vector database. In order to evaluate the performance of different LLMs for generating RAG answers on a specific domain, it is necessary to prepare a sufficiently large list of questions and reference answers. Such a list can be prepared entirely manually by experts in a specific domain. However, this is a slow and time-consuming process. Another widely used approach is to generate relevant questions based on reference answers given by a selected LLM, i.e., creating respective datasets. PaSSER allows the implementation of the second approach.

2. To create a reference dataset for a specific domain, a collection of answers related to the selected domain is collected. Each response should contain key information related to potential queries in that area. These answers should then be saved in a text file format.
3. A selected LLM is deployed to systematically generate a series of questions corresponding to each predefined reference answer. This operation facilitates the creation of a structured dataset comprising pairs of questions and their corresponding answers. Subsequently, this dataset is saved into a JSON file format.
4. The finalized dataset is uploaded to the PaSSER App, initiating an automated sequence of response generation for each query within the target domain. Subsequently, each generated response is forwarded to a dedicated Python backend script. This script is tasked with assessing the responses based on predefined metrics, comparing them to the established reference answers. The outcomes of this evaluation are then stored on the blockchain, ensuring a transparent and immutable ledger of the model's performance metrics.

To facilitate this process, a smart contract 'llmtest' has been created, managing the interaction with the blockchain and providing a structured and secure method for storing and managing the assessment results derived from the LLM performance tests.

The provided code snippet outlines the structure for a *'tests'* table within a blockchain environment, chosen to store test-related entries. It includes identifiers (*id*, *userid*, and *testid*), a timestamp (*created_at*), numerical results (*results* array), and a descriptive text (*description*). It establishes *id* as the primary key for indexing, with additional indices based on *created_at*, *userid*, and *testid* to facilitate data retrieval and sorting by these attributes. This structure organizes and accesses test records within the blockchain.

```
TABLE tests {
    uint64_t id;
    name userid;
    name testid;
    eosio::time_point_sec created_at;
```

 doi:10.20944/preprints202403.0844.v1


```
    std::vector<double> results;
    string description;
    uint64_t primary_key() const { return id; }
    uint64_t third_key() const { return created_at.sec_since_epoch(); }
    uint64_t user_key() const { return userid.value; }
    uint64_t test_key() const { return testid.value; }
  };
```

The code below defines an *eosio::multi_index* structure *tests_table* for a blockchain, which facilitates the storage and indexing of data. It specifies four indices: a primary index based on *id* and secondary indices using *created_at*, *userid*, and *testid* attributes for enhanced query capabilities. These indices optimize data retrieval operations, allowing for efficient access based on different key attributes like timestamp, user, and test identifiers, significantly enhancing the database's functionality within the blockchain environment.

```
  typedef eosio::multi_index<
    name("testtable"),
    tests,
    eosio::indexed_by<
      name("id"),
      eosio::const_mem_fun<
        tests,
        uint64_t,
        &tests::primary_key
      >
    >,
    eosio::indexed_by<
      name("timestamp"),
      eosio::const_mem_fun<
        tests,
        uint64_t,
        &tests::third_key
      >
    >,
    eosio::indexed_by<
      name("users"),
      eosio::const_mem_fun<
        tests,
        uint64_t,
        &tests::user_key
      >
    >,
    eosio::indexed_by<
      name("testid"),
      eosio::const_mem_fun<
        tests,
        uint64_t,
        &tests::test_key
      >
    >
  > tests_table;
```

The provided code defines an EOSIO smart contract action named *add_test*, which allows adding a new record to the *tests_table*. It accepts the creator's name, test ID, description, and an array of results

 doi:10.20944/preprints202403.0844.v1

as parameters. The action assigns a unique ID to the record, stores the current timestamp, and then inserts a new entry into the table using these details. This action helps in dynamically updating the blockchain state with new test information, ensuring that each entry is time-stamped and linked to its creator.

```
    [[eosio::action]] add_test addtest (name creator, name testid, string description,
std::vector<double> results) {
        // require_auth(creator);
        tests_table tests(get_self(), get_self().value);
        uint64_t newid = tests.available_primary_key();
        eosio::time_point_sec timestamp = time_point(current_time_point());
        tests.emplace( get_self(), [&](auto &e) {
          e.id = newid;
          e.userid = creator;
          e.testid = testid;
          e.created_at = timestamp;
          e.description = description;
          e.results = results;
        });
        add_test ret;
        return ret;
      }
```

5. Retrieving the results from the blockchain for further processing and analysis.

To facilitate the execution of this procedures, the interface is structured into three specific submenus: '***Q&A dataset***' for managing question and answer datasets, '***RAG Q&A score test***' for evaluating the performance of RAG utilizing datasets, and '***Show test results***' for displaying the results of the tests. Each submenu is designed to streamline the respective aspect of the workflow, ensuring a coherent and efficient user experience throughout the process of dataset management, performance evaluation, and result visualization.

Within the '***Q&A dataset***' menu, the user is guided to employ a specific prompt, aiming to instruct the LLM to generate questions that align closely with the provided reference answers, as described in step 2. This operation initiates the creation of a comprehensive dataset, subsequently organizing and storing this information within a JSON file for future accessibility and analysis. This approach ensures the generation of relevant and accurate questions, thereby enhancing the dataset's utility for follow-up evaluation processes.

The '***RAG Q&A score test***' menu is designed to streamline evaluating different LLMs' performance using the RAG, as indicated in Figure 5. This evaluation process involves importing a JSON-formatted dataset and linking it with an established vectorstore relevant to the selected domain. The automation embedded within this menu facilitates a methodical assessment of the LLMs, leveraging domain-specific knowledge embedded within the vectorstore.

Vectorstores, once created using a specific LLM's transformers, require the consistent application of the same LLM model during the RAG process. Within this automated framework, each question from the dataset is processed by the LLM to produce a corresponding answer. Then, both the generated answers and its associated reference answer are evaluated by a backend Python script. This script calculates performance metrics, records these metrics on the blockchain under a specified test series, and iterates this procedure for every item within the dataset.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 March 2024 doi:10.20944/preprints202403.0844.v1

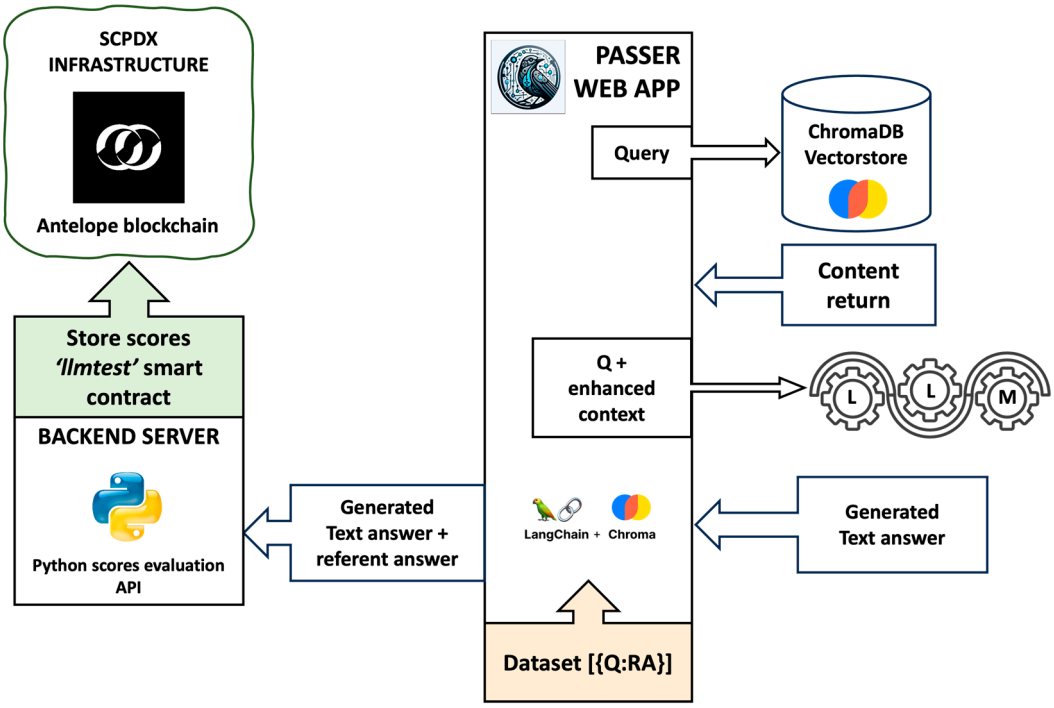


Figure 5. Workflow Diagram for RAG LLM Query Processing and Score Storage.

The '***Show test results***' menu is designed to access and display the evaluation outcomes from various tests as recorded on the blockchain, presenting them in an organized tabular format. This feature facilitates the visualization of score results for individual answers across different test series, and also provides the functionality to export this data into an xlsx file format. The export feature makes it much easier for users to understand and study the data, helping with better evaluations and insights.

The '***Q&A Time LLM Test***' menu evaluates model performance across various hardware setups using JSON-formatted question-answer pairs. Upon submission, PaSSER App prompts the selected model for responses, generating detailed performance metrics like evaluation and load times, among others. These metrics are packed in a query to a backend Python script, which records the data on the blockchain via the *'addtimetest'* action, interacting with the *'llmtest'* smart contract to ensure performance tracking and data integrity.

The '***Show time test results***' menu makes it easy to access and view LLM performance data, organized by test series, from the blockchain. Displayed in a structured table, these metrics can be examined for comprehensive performance assessment. There is an option to export this data into an xlsx file, thereby improving the process for further in-depth examination and analysis.

Authentication within the system ('***Login***') is provided through the Anchor wallet, which is compliant with the security protocols of the SCPDx platform. This process, described in detail in [25], provides user authentication by ensuring that testing activities are securely associated with the correct user credentials. This strengthens the integrity and accountability of the testing process within the platform ecosystem.

The '***Configuration***' feature is divided into '***Settings***' and '***Add Model***' submenus.

The '***Settings***' submenu is designed for configuring connectivity to the Ollama API and ChromaDB API, using IP addresses specified in the application's configuration file. It also allows users to select an LLM from currently installed in Ollama. A key feature here is the ability to adjust the 'temperature' parameter, which ranges from 0 to 1, to fine-tune the balance between creativity and predictability in the output generated by the LLM. Setting a higher temperature value (>0,8) increases randomness, whereas a lower value enhances determinism, with the default set at 0.2.

 doi:10.20944/preprints202403.0844.v1

The '***Add Model***' submenu enables adding and removing LLMs in the Ollama API, allowing dynamic model management. This feature is useful when testing different models, ensuring optimal use of computational resources.

The '***Manage DB***' menu displays a comprehensive list of vectorstores available in ChromaDB, offering functionalities to inspect or interact with specific dataset records. This feature enables users to view details within a record's JSON response. It provides the option to delete any vectorstore that is no longer needed, enabling efficient database management by removing obsolete or redundant data, thereby optimizing storage utilization.

3. Evaluation Metrics

The '***RAG Q&A chat***' is assessed by a set of selected metrics: METEOR, ROUGE, PPL (perplexity), cosine similarity, Pearson Correlation Coefficient, and F1 Score [26]. An automated evaluation framework is developed to apply these metrics to the answers generated using RAG. The framework compares generated answers against the reference answers in the dataset, calculating scores for each metric. Python code is used to calculate the scores. The code is available at GitHub [6] reproducibility.

The following is a brief explanation of the purpose of the metrics used, the simplified calculation formulas, and the application in the context of RAG.

3.1. METEOR (Metric for Evaluation of Translation with Explicit ORdering)

METEOR score is a metric used to assess the quality of machine-generated text by comparing it to one or multiple reference texts [27]. The calculating involves several steps:

- Word alignment between candidate and reference translations based on exact, stem, synonym, and paraphrase matches, with the constraint that each word in the candidate and reference sentences can only be used once and aims to maximize the overall match between the candidate and references.
- Calculation of $Precision\ (P) =$ Number of matched words in the candidate / Number of words in the candidate and $Recall\ (R) =$ Number of matched words in the candidate / Number of words in the reference:

$$P = \frac{m}{w_c}, R = \frac{m}{w_r} \tag{1}$$

Where:

$m =$ Number of unigrams in the candidate translation that are matched with the reference translation.

$w_c =$ Total number of unigrams in the candidate translation.

$w_r =$ Total number of unigrams in the reference translation(s).

- Calculation of *Penalty* for chunkiness which accounts for the arrangement and fluency of the matched chunks *(c)* = Number of chunks of contiguous matched unigrams in the candidate translation and (m):

$$Penalty = 0.5\left(\frac{c}{m}\right)^3 \tag{2}$$

- The final score is computed using the harmonic mean of Precision and Recall, adjusted by the penalty factor:

$$M_{score} = F_{mean}(1 - Penalty) \tag{3}$$

Where $F_{mean} = \frac{10PR}{R+9P}$.

In the context of RAG models, the *METEOR score can be used to evaluate the quality of the generated responses*. A high METEOR score indicates that the generated response closely matches the reference text, suggesting that the model is accurately retrieving and generating responses. Conversely, a low METEOR score could indicate areas for improvement in the model's performance.

 doi:10.20944/preprints202403.0844.v1

3.2. ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

ROUGE [28] is a set of metrics used for evaluating automatic summarization and machine translation. It works by comparing an automatically produced summary or translation against one or more reference summaries (usually human-generated).

The ROUGE has several variants: ROUGE-N, ROUGE-L, and ROUGE-W.

ROUGE-N focuses on the overlap of n-grams (sequences of n words) between the system-generated summary and the reference summaries. It is computed in terms of recall, precision, and F1 score:

– $Recall_{ROUGE-N}$ is the ratio of the number of overlapping n-grams between the system summary and the reference summaries to the total number of n-grams in the reference summaries:

$$Recall_{ROUGE-N} = \frac{\sum_{S\in\{Reference\ Summaries\}}\sum_{gram_n\in s} Count_{match}(gram_n)}{\sum_{S\in\{Reference\ Summaries\}}\sum_{gram_n\in s} Count(gram_n)} \quad (4)$$

– $Precision_{ROUGE-N}$ is the ratio of the number of overlapping n-grams in the system summary to the total number of n-grams in the system summary itself:

$$Precision_{ROUGE-N} = \frac{\sum_{S\in\{System\ Summaries\}}\sum_{gram_n\in s} Count_{match}(gram_n)}{\sum_{S\in\{System\ Summaries\}}\sum_{gram_n\in s} Count(gram_n)} \quad (5)$$

– $F1_{ROUGE-N}$ is the harmonic mean of precision and recall:

$$F1_{ROUGE-N} = 2\frac{Precision_{ROUGE-N}\times Recall_{ROUGE-N}}{Precision_{ROUGE-N}+Recall_{ROUGE-N}} \quad (6)$$

ROUGE-L focuses on the longest common subsequence (LCS) between the generated summary and the reference summaries. The LCS is the longest sequence of words that appears in both texts in the same order, though not necessarily consecutively. The parameters for ROUGE-L include:

- $Recall_{ROUGE-L}$ is a length of the LCS divided by the total number of words in the reference summary. This measures the extent to which the generated summary captures the content of the reference summaries:

$$Recall_{ROUGE-L} = \frac{LCS(System\ Summary, Reference\ Summary)}{Lenght\ of\ Reference\ Summary} \quad (7)$$

- $Precision_{ROUGE-L}$ is a length of the LCS divided by the total number of words in the generated summary. This assesses the extent to which the words in the generated summary appear in the reference summaries:

$$Precision_{ROUGE-L} = \frac{LCS(System\ Summary, Reference\ Summary)}{Lenght\ of\ System\ Summary} \quad (8)$$

- $F1_{ROUGE-L}$ is a harmonic mean of the LCS-based precision and recall:

$$F1_{ROUGE-L} = 2\frac{Precision_{ROUGE-L}\times Recall_{ROUGE-L}}{Precision_{ROUGE-L}+Recall_{ROUGE-L}} \quad (9)$$

ROUGE-W is an extension of ROUGE-L with a weighting scheme that assigns more importance to longer sequences of matching words. In this application ROUGE-W is not applied.

The choice between a preference to precision, recall, or F1-scoring depends on the specific goals of the summarization task, such as whether it is more important to capture as much information as possible (recall) or to ensure that what is captured is highly relevant (precision).

In the context of RAG models, the ROUGE metric serves as a tool for assessing the quality of generated text, especially in summary, question answering and content generation tasks.

3.3. BLEU (Bilingual Evaluation Understudy)

The BLEU [29] score is a metric used to assess the quality of machine-generated text by comparing it to one or multiple reference texts. It quantifies the resemblance by analysing the presence of shared n-grams (sequences of n consecutive words).

 doi:10.20944/preprints202403.0844.v1

The metric employs a combination of modified precision scores for various n-gram lengths and incorporates a brevity penalty to account for the adequacy and fluency of the translation. Below, are the simplified formulas for estimating of BLEU.

For each n-gram length n, BLEU computes a modified precision score. The modified precision P_n for n-grams is calculated as follows:

$$P_n = \frac{\sum_{C\in\{Candidate\ Translation\}}\sum_{n-grams\in C} Count_{clip}(n-gram)}{\sum_{C'\in\{Candidate\ Translation\}}\sum_{n-grams\in C'} Count_{clip}(n-gram')} \quad (10)$$

Where, $Count_{clip}$ is a count of each n-gram in the candidate translation clipped by its maximum count in any single reference translation.

The brevity penalty (BP) is a component of the BLEU score that ensures translations are not only accurate but also of appropriate length. The BP is defined as:

$$BP = \begin{cases} 1 & if\ c > r \\ e^{(1-r/c)} & if\ c \le r \end{cases} \quad (11)$$

Where, c is the total length of the candidate translation, r is the effective reference corpus length, which is the sum of the lengths of the closest matching reference translations for each candidate sentence.

$BP = 1$ if the candidate translation length c is greater than the reference length r, indicating no penalty.

$BP = e^{(1-r/c)}$ if c is less than or equal to r, indicating a penalty that increases as the candidate translation becomes shorter relative to the reference.

The overall BLEU score is calculated using the following formula:

$$BLUE = BP.exp(\sum_{n=1}^{N} w_n log P_n) \quad (12)$$

Where, N is the maximum n-gram length (typically 4), and w_n is the weight for each n-gram's precision score, often set equally such that their sum is 1 (e.g., $w_n = 0.25$ for $N = 4$).

This formula aggregates the individual modified precision scores P_n for n-grams of length 1 to N, geometrically averaged and weighted by w_n, then multiplied by the brevity penalty BP to yield the final BLEU score.

In the context of RAG models, the BLEU score can be used to evaluate the quality of the generated responses. A high BLEU score would indicate that the generated response closely matches the reference text, suggesting that the model is accurately retrieving and generating responses. A low BLEU score could indicate areas for improvement in the model's performance.

3.4. Perplexity (PPL)

Perplexity (PPL) [30] is a measure used to evaluate the performance of probabilistic language models. The introduction of smoothing techniques, such as Laplace (add-one) smoothing and Lidstone smoothing [31], aims to address the issue of zero probabilities for unseen events, thereby enhancing the model's ability to deal with sparse data. Below are the formulas for calculating perplexity.

– PPL with *Laplace Smoothing* adjusts the probability estimation for each word by adding one to the count of each word in the training corpus, including unseen words. This method ensures that no word has a zero probability. Adjusted Probability Estimate with Laplace Smoothing:

$$P_{Laplace}(w_i|h) = \frac{C(w_i,h)+1}{C(h)+V} \quad (13)$$

Where, w_i is the probability of a word given its history h (the words that precede it), $C(w_i, h)$ is the count of w_i, $C(h)$ is the count of history h, V is a vocabulary size (the number of unique words in the training set plus one for unseen words).

The PPL of a sequence of words $W = w_1, \dots w_N$ is given by:

$$PPL(W) = e^{-\frac{1}{N}\sum_{i=1}^{N} ln\left(P_{Laplace}(w_i|h)\right)} \quad (14)$$

 doi:10.20944/preprints202403.0844.v1

– PPL with *Lidstone smoothing* is a generalization of Laplace smoothing where instead of adding one to each count, a fraction λ (where $0 < \lambda < 1$) is added. This allows for more flexibility compared to the fixed increment in Laplace smoothing. Adjusted Probability Estimate with Lidstone Smoothing:

$$P_{Lidstone}(w_i|h) = \frac{C(w_i,h)+\lambda}{C(h)+\lambda V} \quad (15)$$

The PPL of a sequence of words $W = w_1, \dots w_N$ is given by:

$$PPL(W) = e^{-\frac{1}{N}\sum_{i=1}^{N} ln(P_{Lidstone}(w_i|h))} \quad (16)$$

In both formulas, the goal is to compute how well the model predicts the test set W. The lower perplexity indicates that the model predicts the sequence more accurately. The choice between Laplace and Lidstone smoothing depends on the specific requirements of the model and dataset, as well as empirical validation.

In the context of RAG models both metrics are useful for assessing the quality and ability of models to deal with a variety of language and information. These metrics indicate how well they can generate contextually informed, linguistically coherent, and versatile text.

3.5. Cosine Similarity

Cosine similarity [32] is a measure of vector similarity, and can be used to determine the distance of embeddings between the chunk and the query. It is a distance metric that approaches 1 when the question and chunk are similar, and becomes 0 when they are different. The mathematics formulation of the metric is:

$$Cosin\ Similarity = \frac{A.B}{\|A\|\,\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \quad (17)$$

Where, $A.B$ is the dot product of vectors A and B, $\|A\|$ and $\|B\|$ are the Euclidean norms (magnitudes) of vectors A and B, calculated $\sqrt{\sum_{i=1}^{n} A_i^2}$ and $\sqrt{\sum_{i=1}^{n} B_i^2}$ respectively and n is the dimensionality of the vectors, assuming A and B of the same dimension.

$Cosin\ Similarity = 1$ means the vectors are identical in orientation.

$Cosin\ Similarity = 0$ means the vectors are orthogonal (independent) to each other.

$Cosin\ Similarity = -1$ means the vectors are diametrically opposed.

In the RAG models, cosine similarity ensures that retrieved documents align closely with user queries, capturing relationships between the meaning of a user. This is particularly important in RAG models, as they leverage a retriever to find context documents. The use of cosine similarity between embeddings ensures that these retrieved documents align closely with user queries.

3.6. Pearson Correlation

Pearson Correlation Coefficient *(r)* is a statistical measure that calculates the strength and direction of the linear relationship between two continuous variables.

The formula for the Pearson correlation coefficient is as follows:

$$r = \frac{\sum_{i=1}^{n}(X_i-\bar{X})(Y_i-\bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i-\bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i-\bar{Y})^2}} \quad (18)$$

Where, n is the number of data points, X_i and Y_i are the individual data points, $\bar{X}$ and $\bar{Y}$ are the means of the X and Y data sets, respectively.

In the context of evaluating RAG models, the Pearson Correlation Coefficient can be used to measure how well the model's predictions align with actual outcomes. A coefficient close to +1 indicates a strong positive linear relationship, meaning as one variable increases, the other also increases. A coefficient close to -1 indicates a strong negative linear relationship, meaning as one variable increases, the other decreases. A coefficient near 0 suggests no linear correlation between variables. In the evaluation of RAG models, a high Pearson Correlation Coefficient could indicate

 doi:10.20944/preprints202403.0844.v1

that the model is accurately retrieving and generating responses, while a low coefficient could suggest areas for improvement.

3.7. F1 Score

In the context of evaluating the performance of RAG models, the F1 score [33] is used for quantitatively assessing how well the models perform in tasks for generating or retrieving textual information (question answering, document summarization, or conversational AI). The evaluation often hinges on their ability to accurately and relevantly generate text that aligns with reference or ground truth data.

The F1 score is the harmonic mean of precision and recall. *Precision* assesses the portion of relevant information in the responses generated by the RAG model. High precision indicates that most of the content generated by the model is relevant to the query or task at hand, minimizing irrelevant or incorrect information. *Recall* (or sensitivity) evaluates the model's ability to capture all relevant information from the knowledge base that should be included in the response. High recall signifies that the model successfully retrieves and incorporates a significant portion of the pertinent information available in the context.

The formula for calculating is:

$$\mathrm{F1} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{19}$$

Precision and Recall are defined as:

$$Precision = \frac{TP}{TP+FP}, Recall = \frac{TP}{TP+FN} \tag{20}$$

Where, TP (True Positives) is the count of correctly retrieved relevant documents, FP (False Positives) is the count of incorrectly retrieved documents (i.e., the documents that were retrieved but are not relevant), and FN (False Negatives) is the count of relevant documents that were not retrieved.

For tasks of question answering, the F1 score can be used to measure how well the generated answers match the expected answers, considering both the presence of correct information (high precision) and the completeness of the answer (high recall).

For tasks of document summarization, the F1 score might evaluate the overlap between the key phrases or sentences in the model-generated summaries and those in the reference summaries, reflecting the model's efficiency in capturing essential information (recall) and avoiding extraneous content (precision).

For, conversational AI applications, the F1 score could assess the relevance and completeness of the model's responses in dialogue, ensuring that responses are both pertinent to the conversation context and comprehensive in addressing users' intents or questions.

4. Testing

The aim of the tests presented in this section is to evaluate the performance of the Misrtal:7b, Llama2:7b, and Orca2:7b models installed on two different hardware configurations and to assess the performance of these models in generating answers using RAG on the selected knowledge domain, smart agriculture.

The knowledge base used is retrieved form EU Regulation 2018/848 (https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32018R0848) and Climate-smart agriculture Sourcebook (https://www.fao.org/3/i3325e/i3325e.pdf). These documents are pre-processed manually and vectorized using the transformers of Misrtal:7b, Llama2:7b, and Orca2:7b LLMs under the parameters: chunk size - 500, overlapping - 100, temperature - 0.2.

The dataset containing reference answers specific to smart agriculture is compiled and stored as a text file. The Mistral:7b model is deployed to formulate questions based on these reference answers. Initial trials indicate that Mistral:7b excels in generating questions with high relevance within this particular domain. To initiate the question generation process, a specifically crafted prompt is

 doi:10.20944/preprints202403.0844.v1

employed: "Imagine you are a virtual assistant trained in the detailed regulations of organic agriculture. Your task involves creating precise question for a specific regulatory statement provided to you below. The statement comes directly from the regulations, and your challenge is to reverse-engineer the question that this statement answers. Your formulated question should be concise, clear, and directly related to the content of the statement. Aim to craft your question without implying the statement itself as the answer, and, where relevant, gear your question towards eliciting specific events, facts, or regulations."

For testing purposes, Ollama API is installed on the following hardware configurations:

- Intel Xeon, 32 Cores, 128 GB RAM, Ubuntu 22.04 w/o GPU.
- Mac M1, 8 CPU, 10 GPU, 16GB RAM, OSX 13.4.

For this purpose, the two following tests are designed: test via features '***Q&A Time LLM Test***' and '***RAG Q&A score test***'.

The '***Q&A Time LLM Test***' evaluates LLM performance across two hardware configurations using a dataset of 446 questions for each model, focusing on specific 7 metrics (evaluation time, evaluation count, load duration time, prompt evaluation count, prompt evaluation duration, total duration, and tokens per second. These metrics are integral for analyzing the efficiency and responsiveness of each model under different computational conditions. The collected data is stored on a blockchain, ensuring both transparency and traceability of the evaluation results.

The '***RAG Q&A score test***' aims to evaluate the performance of the models based on 13 metrics (METEOR, ROUGE-1, ROUGE-l, BLEU, perplexity, cosine similarity, Pearson correlation and F1) applied to each of the 446 question – reference answers – RAG obtained answers.

The '***RAG Q&A score test***' evaluates the performance of different models in a chat environment with enhanced RAG Q&A, identifying differences and patterns in their ability to respond to queries. Its goal is to determine the model that best provides accurate, context-aware responses, tops at defining terms, and summarizes specific content. This evaluation can be used to select a model that ensures the delivery of accurate and relevant information in the context of the specific knowledge provided.

The performance outcomes from the '***Q&A Time LLM Test***' and '***RAG Q&A score test***' for evaluating LLMs are stored on the blockchain via smart contracts. For analysis, this data is retrieved from the blockchain and stored in an xlsx file. This file is uploaded to GitHib (https://github.com/scpdxtest/PaSSER/blob/main/tests/TEST%20DATA_GENERAL%20FILE.xlsx).

In the upcoming section, the focus is solely on presenting and analysing the mean values derived from the test data. This approach eases the interpretation, enabling a summarized review of the core findings and trends across the conducted evaluations.

4.1. Q&A Time LLM Test Results

When using the '***Q&A Chat***' feature, performance metrics are returned by the Ollama API along with each response. The metrics are reported in nanoseconds and converted to seconds for easier interpretation. They are used to evaluate the performance of different models under different hardware configurations.

Evaluation count is a number of individual evaluations performed by the model during a specific test. It helps to understand it's capacity to process multiple prompts or tasks within a given timeframe, reflecting its productivity under varying workloads.

Load duration time is the time required for initializing and loading the language model into memory, before it begins processing data. This metric is essential for assessing the start-up efficiency of the model, which can significantly impact user experience and operational latency in real-time applications.

Prompt evaluation count is the number of prompts the model evaluates within a specific test or operational period. It provides insights into the model's interactive performance, especially relevant in scenarios where the model is expected to respond to user inputs or queries dynamically.

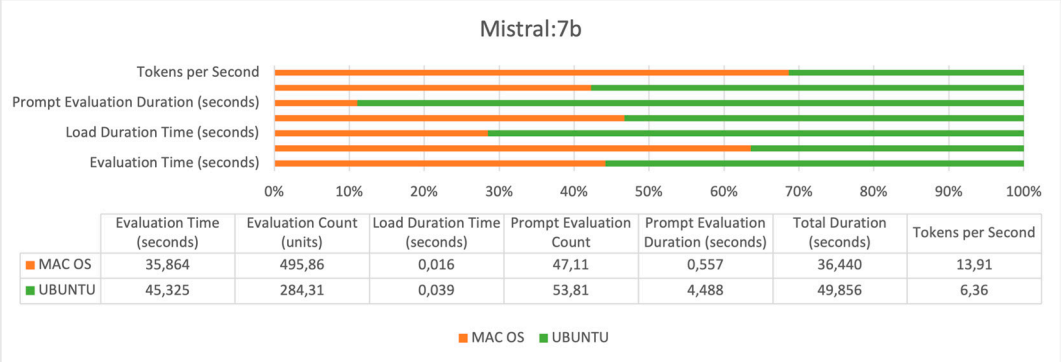
Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 March 2024 doi:10.20944/preprints202403.0844.v1

Prompt evaluation duration captures the time model uses to evaluate a single prompt, from receiving the input to generating the output. It measures the model's responsiveness and is particularly important for interactive applications where fast output is vital.

Total duration is the overall time for the entire evaluation process, incorporating all phases from initialization to the completion of the last prompt evaluation. This metric gives a complete view of the model's operational efficiency over an entire test cycle.

Tokens per second quantifies the number of tokens (basic units of text, such as words or characters) the model can process per second. It is a key indicator of the model's processing speed and computational throughput, reflecting its ability to handle large volumes of text data efficiently.

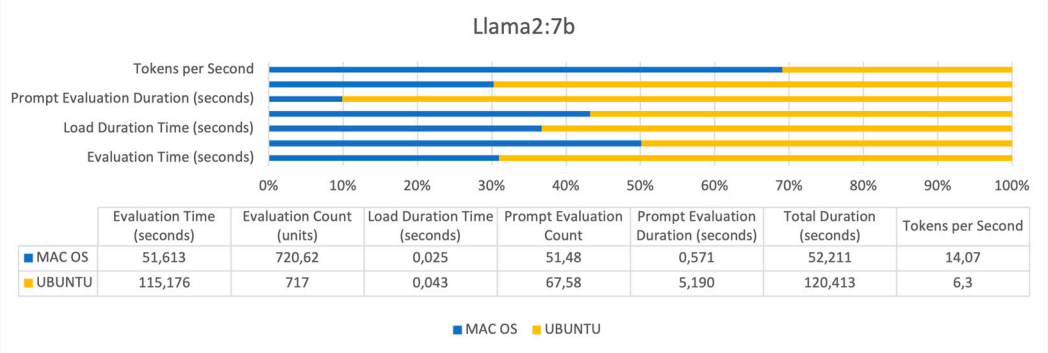
The Figure 6 illustrates the performance metrics of the Mistral:7b on macOS and Ubuntu operating systems across the different indicators. The model demonstrates higher efficiency on macOS, as indicated by the shorter evaluation time, longer prompt evaluation count, and significantly higher tokens per second rate. Conversely, the model takes longer to process prompts on Ubuntu, as indicated by the extended duration of prompt evaluation, even though the operating system manages a greater number of prompt evaluations overall..



	Evaluation Time (seconds)	Evaluation Count (units)	Load Duration Time (seconds)	Prompt Evaluation Count	Prompt Evaluation Duration (seconds)	Total Duration (seconds)	Tokens per Second
MAC OS	35,864	495,86	0,016	47,11	0,557	36,440	13,91
UBUNTU	45,325	284,31	0,039	53,81	4,488	49,856	6,36

Figure 6. A performance of Mistral:7b.

The Figure 7 presents a comparison of the Llama2:7b performance. It reveals MAC OS as the more efficient platform, processing higher number of tokens per second and completing evaluations faster. Despite Ubuntu's slightly higher prompt evaluation count, it shows longer prompt evaluation and total duration times.

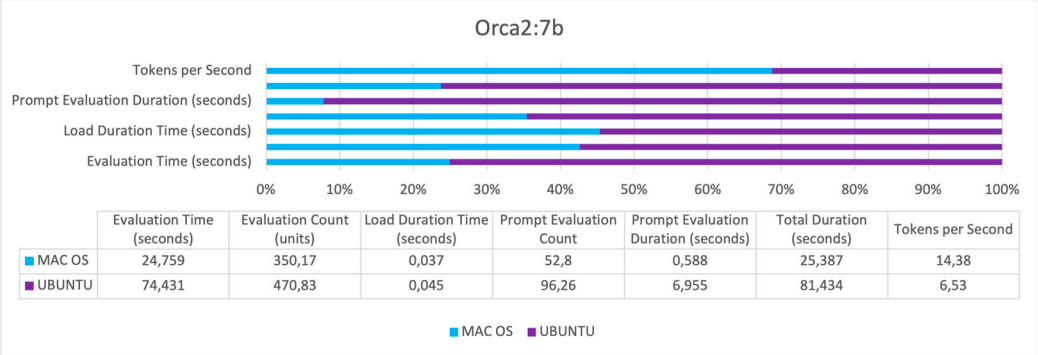


	Evaluation Time (seconds)	Evaluation Count (units)	Load Duration Time (seconds)	Prompt Evaluation Count	Prompt Evaluation Duration (seconds)	Total Duration (seconds)	Tokens per Second
MAC OS	51,613	720,62	0,025	51,48	0,571	52,211	14,07
UBUNTU	115,176	717	0,043	67,58	5,190	120,413	6,3

Figure 7. A performance of Llama2:7b.

The Figure 8 illustrates performance metrics for the Orca2:7b. MAC OS outperforms Ubuntu with faster evaluation times, shorter load durations, and notably higher tokens per second processing efficiency. While Ubuntu managed a higher prompt evaluation count, it lagged in prompt evaluation speed, as reflected in the longer prompt evaluation duration and extended total duration times.

 doi:10.20944/preprints202403.0844.v1



	Evaluation Time (seconds)	Evaluation Count (units)	Load Duration Time (seconds)	Prompt Evaluation Count	Prompt Evaluation Duration (seconds)	Total Duration (seconds)	Tokens per Second
MAC OS	24,759	350,17	0,037	52,8	0,588	25,387	14,38
UBUNTU	74,431	470,83	0,045	96,26	6,955	81,434	6,53

Figure 8. A performance of Orca2:7b.

Across all three models, several trends are evident (Figures 6–8). UBUNTU generally shows longer *evaluation times* across all models, indicating slower processing capabilities compared to MAC OS. *Evaluation counts* are relatively comparable, suggesting that the number of operations conducted within a given timeframe is similar across hardware configurations. *Load duration times* are consistently longer on UBUNTU, affecting readiness and response times negatively. UBUNTU tends to conduct more *prompt evaluation count*, but also takes significantly longer, which exposes efficiency issues. UBUNTU experiences longer *total durations* for all tasks, reinforcing the trend of slower overall performance. MAC OS demonstrates higher *tokens per second* across all models, indicating more efficient data processing capabilities.

The performance indicators presented in Table 1 suggest that across all three models, the evaluation time on the Mac M1 system is significantly less than on the Ubuntu system with Xeon processors, indicating faster overall performance. In terms of tokens per second, the Mac M1 also performs better, suggesting it is more efficient at processing information regardless of having fewer CPU cores and less RAM.

Table 1. Comparative Performance Metrics of Llama2:7b, Mistral:7b, and Orca2:7b Language Models on macOS M1 and Ubuntu Xeon Systems **(w/o GPU)**.

	Llama2:7b		Mistral:7b		Orca2:7b	
Metric	macOS / M1	Ubuntu / Xeon	macOS / M1	Ubuntu / Xeon	macOS / M1	Ubuntu / Xeon
Evaluation Time (sec.)	Faster (51,613)	Slower (115,176)	Faster (35,864)	Slower (45,325)	Fastest (24,759)	Slowest (74,431)
Evaluation Count (units)	Slightly Higher (720)	Comparable (717)	Higher (496)	Lower (284)	Lower (350)	Higher (471)
Load Duration Time (sec.)	Faster (0.025)	Slower (0.043)	Fastest (0.016)	Slower (0.039)	Similar (0.037)	Similar (0.045)
Prompt Evaluation Count	Lower (51)	Higher (68)	Lower (47)	Higher (54)	Lower (53)	Highest (96)
Prompt Evaluation Duration (sec.)	Shorter (0.571)	Longer (5.190)	Shorter (0.557)	Longer (4.488)	Shorter (0.588)	Longest (6.955)
Total Duration (sec.)	Shorter (52,211)	Longer (120,413)	Shorter (36,440)	Longer (49,856)	Shortest (25,387)	Longer (81,434)
Tokens/Second	Higher (14.07)	Lower (6.3)	Higher (13.91)	Lower (6.36)	Highest (14.38)	Lower (6.53)

Despite having a higher core count and more RAM, the evaluation time is longer, and the tokens per second rate is lower on the Ubuntu system. This suggests that the hardware advantages of the Xeon system are not translating into performance gains for these particular models. Notably, the

 doi:10.20944/preprints202403.0844.v1

Ubuntu system shows a higher prompt evaluation count for Orca2:7b, which might be leveraging the greater number of CPU cores to handle more prompts simultaneously.

Orca2:7b has the lowest evaluation time on the Mac M1 system, showcasing the most efficient utilization of that hardware. Llama2:7b shows the significant difference in performance between the two systems, indicating it may be more sensitive to hardware and operating system optimizations. Mistral:7b has a comparatively closer performance between the two systems, suggesting it may be more adaptable to different hardware configurations.

The table suggests that the Mac M1's architecture provides a significant performance advantage for these language models over the Ubuntu system equipped with a Xeon processor. This could be due to several factors, including but not limited to, the efficiency of the M1 chip, the optimization of the language models for the specific architectures, and the potential use of the M1's GPU in processing.

4.2. RAG Q&A Score Test Results

In this section, the performance of the Mistral:7b, Llama2:7b and Orca2:7b models through several key metrics: METEOR, ROUGE-1, ROUGE-L, BLEU, perplexity, cosine similarity, Pearson correlation coefficient and F1 score is assessed. A summary of the average metrics is presented in Table 2.

Table 2. A summary of performance metrics using RAG Q&A Chat.

Metric	*Llama2:7b*	*Mistral:7b*	*Orca2:7b*	*Best Model*	*Metric in Text Generation and Summarization tasks*
METEOR	0.248	***0.271***	0.236	Mistral:7b	Assesses fluency and adequacy of generated text response, considering synonymy and paraphrase.
ROUGE-1 recall	0.026	***0.032***	0.021	Mistral:7b	Measures the extent to which a generated summary captures key points from a source text, indicating coverage.
ROUGE-1 precision	0.146	***0.161***	0.122	Mistral:7b	Evaluates the fraction of content in the generated summary that is relevant to the source text, implying conciseness.
ROUGE-1 f-score	0.499	0.472	***0.503***	Orca2:7b	Provides a balance between recall and precision for assessing the overall quality of a generated summary.
ROUGE-l recall	0.065	***0.07***	0.055	Mistral:7b	Reflects the degree to which a generated lowercase summary encompasses the content of a reference lowercase summary.
ROUGE-l precision	0.131	***0.143***	0.108	Mistral:7b	Measures the accuracy of a generated lowercase summary in replicating the significant elements of the source text.
ROUGE-l f-score	0.455	0.424	***0.457***	Orca2:7b	Integrates precision and recall to evaluate the quality of a generated lowercase summary holistically.
BLUE	0.186	***0.199***	0.163	Mistral:7b	Quantifies the similarity of the generated text to reference texts by comparing n-grams, useful for machine translation and summarization.
Laplace Perplexity	***52.992***	53.06	53.083	Llama2:7b	Estimates the likelihood of a sequence in generated text, indicating how well the text generation model predicts sample sequences.

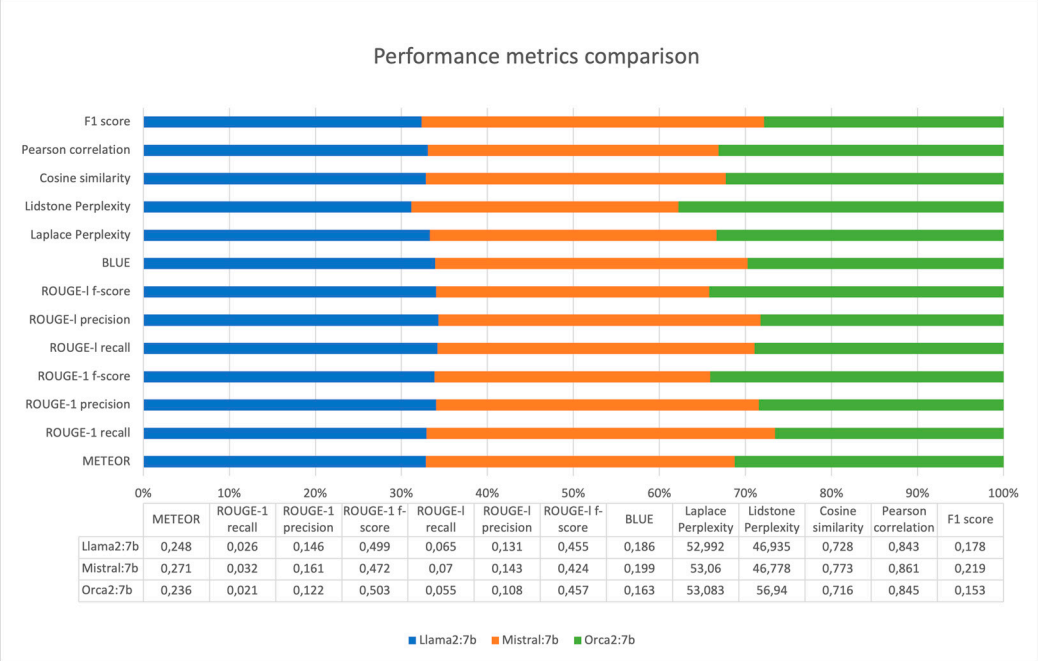
 doi:10.20944/preprints202403.0844.v1

Metric	*Llama2:7b*	*Mistral:7b*	*Orca2:7b*	*Best Model*	*Metric in Text Generation and Summarization tasks*
Lidstone Perplexity	***46.935***	46.778	56.94	Mistral:7b	Assesses the smoothness and predictability of a text generation model by evaluating the likelihood of sequence occurrence with small probability adjustments.
Cosine similarity	***0.728***	0.773	0.716	Mistral:7b	Determines the semantic similarity between the vector representations of generated text and reference texts.
Pearson correlation	***0.843***	0.861	0.845	Mistral:7b	Quantifies the linear correspondence between generated text scores and human-evaluated scores, indicating model predictability and reliability.
F1 score	***0.178***	0.219	0.153	Mistral:7b	Combines the precision and recall of the generated text in summarization tasks, providing a singular measure of its informational quality.

For a more straightforward interpretation of the results, the ranges of values of the different metrics are briefly described below.

The *ideal METEOR score is 1*. It indicates a perfect match between the machine-generated text and the reference translations, encompassing both semantic and syntactic accuracy. For ROUGE metrics (*ROUGE-1 recall, precision, f-score, ROUGE-l recall, precision, f-score), the best possible value is 1*. This value denotes a perfect overlap between the content generated by the model and the reference content, indicating high levels of relevance and precision in the captured information. *The BLEU score's maximum is also 1* (or 100 when expressed in percentage terms), representing an exact match between the machine's output and the reference texts, reflecting high coherence and context accuracy. *For Perplexity, the lower the value, the better the model's predictive performance*. The best perplexity score would technically approach 1, indicating the model's predictions are highly accurate with minimal uncertainty. *The Cosine similarity of 1 signifies maximum similarity between the generated output and the reference*. *A Pearson correlation of 1 is ideal*, signifying a perfect positive linear relationship between the model's outputs and the reference data, indicating high reliability of the model's performance. *An F1 score reaches its best at 1*, representing perfect precision and recall, meaning the model has no false positives or false negatives in its output. For the better comparison of the models Figure 9 is presented.

 doi:10.20944/preprints202403.0844.v1



	METEOR	ROUGE-1 recall	ROUGE-1 precision	ROUGE-1 f-score	ROUGE-l recall	ROUGE-l precision	ROUGE-l f-score	BLUE	Laplace Perplexity	Lidstone Perplexity	Cosine similarity	Pearson correlation	F1 score
Llama2:7b	0,248	0,026	0,146	0,499	0,065	0,131	0,455	0,186	52,992	46,935	0,728	0,843	0,178
Mistral:7b	0,271	0,032	0,161	0,472	0,07	0,143	0,424	0,199	53,06	46,778	0,773	0,861	0,219
Orca2:7b	0,236	0,021	0,122	0,503	0,055	0,108	0,457	0,163	53,083	56,94	0,716	0,845	0,153

Figure 9. Performance metrics comparison.

The presented metrics provide a picture of the performance of the models on text generation and summarization tasks. The analysis for each metric is as follows.

METEOR evaluates the quality of translation by aligning the model output to reference translations considering precision and recall. *Mistral:7b scores highest, suggesting its translations or generated text are the most accurate.*

ROUGE-1 Recall measures the overlap of unigrams between the generated summary and the reference. A higher score indicates more content overlap. *Mistral:7b leads, which implies it includes more of the reference content in its summaries or generated text.*

ROUGE-1 Precision (the unigram precision). Mistral:7b has the highest score, indicating that its content is more relevant and has fewer irrelevant inclusions.

ROUGE-1 F-Score is the harmonic mean of precision and recall. Orca2:7b leads slightly, indicating a balanced trade-off between precision and recall in its content generation.

ROUGE-L Recall measures the longest common subsequence and is good at evaluating sentence-level structure similarity. *Mistral:7b scores the highest, showing it is better at capturing longer sequences from the reference text.*

ROUGE-L Precision. Mistral:7b again scores highest, indicating it includes longer, relevant sequences in its summaries or generated text without much irrelevant information.

ROUGE-L F-Score. Orca2:7b has a marginally higher score, suggesting a balance in precision and recall for longer content blocks.

8. BLEU assesses the quality of machine-generated translation. Mistral:7b outperforms the others, indicating its translations may be more coherent and contextually appropriate.

Laplace Perplexity. For perplexity, a lower score is better as it indicates a model's predictions are more certain. *Llama2:7b has the lowest score, suggesting the best predictability under Laplace smoothing conditions.*

Lidstone Perplexity - Mistral:7b has the lowest score, indicating it is slightly more predictable under Lidstone smoothing conditions.

Cosine Similarity measures the cosine of the angle between two vectors. A higher score indicates greater semantic similarity. *Mistral:7b has the highest score, suggesting its generated text is most similar to the reference text in terms of meaning.*

 doi:10.20944/preprints202403.0844.v1

Pearson Correlation measures the linear correlation between two variables. A score of 1 indicates perfect correlation. *Mistral:7b has the highest score, showing its outputs have a stronger linear relationship with the reference data.*

F1 Score balances precision and recall. Mistral:7b has the highest F1 score, indicating the best balance between recall and precision in its outputs.

Based on the above analysis, the following summary can be concluded. For text generation and summarization, Mistral:7b appears to be the best performing model in most metrics, particularly those related to semantic quality and relevance. Orca2:7b shows strength in balancing precision and recall, especially for longer content sequences. Llama2:7b demonstrates the best predictive capability under Laplace smoothing conditions, which may be beneficial in certain predictive text generation tasks. The selection of the best model would depend on the specific requirements of the text generation or summarization task.

5. Discussion

The PaSSER App testing observations reveal several aspects that affect the acquired results, the performance, and can be managed. These are data cleaning and pre-processing, chunk sizes, GPU usage, and RAM size.

Data cleaning and pre-processing cannot be fully automated. In addition to removing special characters and hyperlinks, it is also necessary to remove non-essential or erroneous information, standardize formats and correct errors from the primary data. This is done manually. Because at this stage PaSSER processes only textual information, the normalisation of data, handling of missing data, detection and removal of deviations is not considered.

Selecting documents with current, validated, and accurate data is pivotal, yet this process cannot be entirely automated. What can be achieved, is to ensure traceability and record the updates and origins of both primary and processed data, along with their secure storage. Blockchain and distributed file systems can be used for this purpose. Here, this objective is partially implemented since blockchain is used solely to record the testing results..

The second aspect is chunk sizes when creating and using vectorstores. Smaller chunks require less memory and computational resources. This is at the expense of increased iterations and overall execution time, which is balanced by greater concurrency in query processing. On the other hand, larger chunks provide more context, but may be more demanding on resources and potentially slow down processes if not managed efficiently.

Adjusting the chunk size affects both the recall and precision of the results. Adequate chunk size is essential to ensure a balance between the retrieval and generation tasks in RAG, as over- or undersized chunks can negatively impact one or both components. In the tests, 500 character chunk sizes were found to give the best results. In this particular implementation, no metadata added (document type or section labels) is used in the vectorstore creation process, which would facilitate more targeted processing when using smaller chunks.

GPU usage and RAM size obviously affect the performance of the models. It is evident from the results that hardware configurations that do not use the GPU perform significantly slower on the text generation and summarization tasks. Models with fewer parameters (up to 13b) can run reasonably well on 16GB RAM configurations. Larger models need more resources, both in terms of RAM and GPU resources. This is the reason, in this particular implementation, to use the selected small LLMs, which are suitable for standard and commonly available hardware configurations and platforms.

It is important to note that the choice of a model may vary depending on the specific requirements of a given task, including the desired balance between creativity and accuracy, the importance of fluency versus content fidelity, and the computational resources available. Therefore, while Mistral:7b appears to be the most versatile and capable model based on the provided metrics, the selection of a model should be guided by the specific objectives and constraints of the application in question.

 doi:10.20944/preprints202403.0844.v1

6. Conclusion

This paper presented the development, technology integration and use of the PaSSER web application, designed to leverage RAG technology and LLMs for enhanced document retrieval and analysis. The web application incorporates various technologies: JavaScript and Primereact for developing the web interface, Blockchain (Antelope) for secure and verifiable data transactions, ChromaDB for utilizing vector database to manage, and retrieve data efficiently, supporting semantic search capabilities, LangChain Library and Ollama API for interaction with LLMs and RAG technology.

The web application integrates three large language models, Mistral:7b, Llama2:7b:

Orca2:7b, selected for their performance and compatibility with medium computational capacity hardware. It has built-in testing modules that evaluate the performance and efficiency of integrated LLMs in real-time, enabling application improvement based on empirical data. Although it is specifically highlighted that the chosen generality of specialized knowledge is smart agriculture, PaSSER's flexible architecture and capabilities make it suitable for any domain. The web application employs a set of 13 evaluation metrics (ROUGE-1 recall, precision, f-score; ROUGE-l recall, precision, f-score; BLUE, Laplace Perplexity, Lidstone Perplexity, Cosine similarity, Pearson correlation, F1 score) to assess the performance of its integrated LLMs within the RAG framework.

Two primary tests have been conducted to evaluate the LLMs. The **'Q&A Time LLM Test'** were focused on assessing the performance of LLMs across different hardware configurations. The ***'RAG Q&A Score Test'*** applied the set of selected metrics to a created dataset of 446 question-answer pairs curated for the smart agriculture domain, aiming to determine which model delivers the most accurate and contextually relevant responses within the RAG framework. The evaluation showcased the strengths of each model in various aspects of text generation and summarization, with specific metrics helping to identify areas of excellence and potential improvement for each LLM.

The tests conducted within the PaSSER, particularly the ***'RAG Q&A Score Test'***, provided a comparative analysis across various performance metrics. While the detailed comparative results for each model are outlined, the tests identify the Mistral:7b model as exhibiting superior performance in several key areas. This model demonstrated higher scores in metrics that evaluate the fluency, adequacy, and overall quality of text generation, such as METEOR and various ROUGE metrics. The model effectively generated responses that were closely aligned with the context and content of the reference data, as indicated by its performance in cosine similarity and Pearson correlation metrics. In tasks requiring a nuanced balance between capturing relevant information and minimizing irrelevant content, Mistral:7b showed best performance, particularly reflected in its F1 scores.

These results suggest that Mistral:7b was the most adept among the evaluated models at synthesizing and articulating information in a manner that is contextually pertinent and linguistically coherent, making it the preferred choice based on the specific evaluation criteria and testing environment outlined in the PaSSER application's assessment.

The future development of the PaSSER App will be focused in the following directions. Leveraging new pre-trained open-source LLMs (over 40b) and testing for RAG scoring and performance. Conducting experiments with different finetuning approaches and developing a procedure for recording results in the blockchain. Integrate PaSSER functionalities into the SCPDx platform in terms of domain-specific knowledge retrieval and processing, and storage and exchange of pre-processed data and respective datasets in the existing Antelope blockchain/IPFS infrastructure.

Author Contributions: Conceptualization, I.R. and I.P.; methodology, I.R. and I.P.; software, I.R, M.D.; validation, I.R, M.D; formal analysis, I.P. and I.R.; investigation, L.D.; resources, L.D.; data curation, I.R. and M.D.; writing—original draft preparation, I.R. and I.P; writing—review and editing, I.R., I.P, M.D.; visualization, I.R., M.D.; supervision, I.P.; project administration, L.D.; funding acquisition, I.R, I.P., and L.D. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Bulgarian Ministry of Education and Science under the National Research Program "Smart crop production" approved by the Ministry Council No. 866/26.11.2020.

Data Availability Statement: All data and source codes are available at: https://github.com/scpdxtest/PaSSER. Git Structure: `*README.md*` - information about the project and instructions on how to use it; `*package.json*`- the list of project dependencies and other metadata; `*src*`- all the source code for the project; `*src/components*` - all the React components for the project; `*src/components/configuration.json*` - various configuration options for the app; `*src/App.js*` - the main React component that represents the entire app; `*src/index.js*` - JavaScript entry point file; `*public*` - static files like the `index.html` file; `*scripts*` - Python backend scripts; `*Installation Instructions.md*` - contains instructions on how to install and set up the project.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Hambarde, K.A.; Proença, H. Information Retrieval: Recent Advances and Beyond. *IEEE Access* **2023**, *11*, 76581–76604, doi:10.1109/ACCESS.2023.3295776.
2. Sidorov, G. Vector Space Model for Texts and the Tf-Idf Measure. In *Syntactic n-grams in Computational Linguistics*; Sidorov, G., Ed.; Springer International Publishing: Cham, 2019; pp. 11–15 ISBN 978-3-030-14771-6.
3. Jelodar, H.; Wang, Y.; Yuan, C.; Feng, X.; Jiang, X.; Li, Y.; Zhao, L. Latent Dirichlet Allocation (LDA) and Topic Modeling: Models, Applications, a Survey. *Multimedia Tools and Applications* **2019**, *78*, 15169–15211, doi:10.1007/s11042-018-6894-4.
4. Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.; Rocktäschel, T.; et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks 2021.
5. Gao, Y.; Xiong, Y.; Gao, X.; Jia, K.; Pan, J.; Bi, Y.; Dai, Y.; Sun, J.; Guo, Q.; Wang, M.; et al. Retrieval-Augmented Generation for Large Language Models: A Survey 2024.
6. GitHub - Scpdxtest/PaSSER Available online: https://github.com/scpdxtest/PaSSER (accessed on 8 March 2024).
7. Popchev, I.; L. Doukovska; I. Radeva A Framework of Blockchain/IPFS-Based Platform for Smart Crop Production. In Proceedings of the International Conference Automatics and Informatics; IEEE: Varna, Bulgaria, October 6 2022; pp. 265–270.
8. Popchev, I.; L. Doukovska; I. Radeva A Prototype of Blockchain/Distributed File System Platform.; IEEE: Warsaw, Poland, October 12 2022; pp. 1–7.
9. Ilieva, G.; Yankova, T.; Radeva, I.; Popchev, I. Blockchain Software Selection as a Fuzzy Multi-Criteria Problem. *Computers* **2021**, *10*, doi:10.3390/computers10100120.
10. Radeva, I.; I. Popchev Blockchain-Enabled Supply-Chain in Crop Production Framework. *Cybernetics and Information Technologies* **2022**, *22*, 151–170, doi:10.2478/cait-2022-0010.
11. Popchev, I.; Radeva, I.; Doukovska, L. Oracles Integration in Blockchain-Based Platform for Smart Crop Production Data Exchange. *Electronics* **2023**, *12*, 2244, doi:10.3390/electronics12102244.
12. GitHub - Chroma-Core/Chroma: The AI-Native Open-Source Embedding Database Available online: https://github.com/chroma-core/chroma (accessed on 26 February 2024).
13. NLTK :: Natural Language Toolkit Available online: https://www.nltk.org/ (accessed on 26 February 2024).
14. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems 32* 2019, 8024–8035.
15. NumPy Documentation — NumPy v1.26 Manual Available online: https://numpy.org/doc/stable/ (accessed on 26 February 2024).
16. pltrdy Rouge: Full Python ROUGE Score Implementation (Not a Wrapper).
17. contributors (https://github.com/huggingface/transformers/graphs/contributors), T.H.F. team (past and future) with the help of all our Transformers: State-of-the-Art Machine Learning for JAX, PyTorch and TensorFlow.
18. SciPy Documentation — SciPy v1.12.0 Manual Available online: https://docs.scipy.org/doc/scipy/ (accessed on 26 February 2024).
19. Pyntelope Available online: https://pypi.org/project/pyntelope/ (accessed on 27 February 2024).
20. Rastogi, R. Papers Explained: Mistral 7B. *DAIR.AI* 2023.
21. Mistral 7B Available online: https://ar5iv.labs.arxiv.org/html/2310.06825 (accessed on 6 March 2024).

22. Workers AI Update: Hello, Mistral 7B! Available online: https://blog.cloudflare.com/workers-ai-update-hello-mistral-7b (accessed on 6 March 2024).
23. Meta-Llama/Llama-2-7b · Hugging Face Available online: https://huggingface.co/meta-llama/Llama-2-7b (accessed on 6 March 2024).
24. Mitra, A.; Corro, L.D.; Mahajan, S.; Codas, A.; Ribeiro, C.S.; Agrawal, S.; Chen, X.; Razdaibiedina, A.; Jones, E.; Aggarwal, K.; et al. Orca-2: Teaching Small Language Models How to Reason. **2023**.
25. Popchev, I.; Radeva, I.; Dimitrova, M. Towards Blockchain Wallets Classification and Implementation. In Proceedings of the 2023 International Conference Automatics and Informatics (ICAI); October 2023; pp. 346–351.
26. Chen, J.; Lin, H.; Han, X.; Sun, L. Benchmarking Large Language Models in Retrieval-Augmented Generation 2023.
27. Banerjee, S.; Lavie, A. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In Proceedings of the Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization; Goldstein, J., Lavie, A., Lin, C.-Y., Voss, C., Eds.; Association for Computational Linguistics: Ann Arbor, Michigan, June 2005; pp. 65–72.
28. Lin, C.-Y. ROUGE: A Package for Automatic Evaluation of Summaries. In Proceedings of the Text Summarization Branches Out; Association for Computational Linguistics: Barcelona, Spain, July 2004; pp. 74–81.
29. Papineni, K.; Roukos, S.; Ward, T.; Zhu, W.-J. Bleu: A Method for Automatic Evaluation of Machine Translation. In Proceedings of the Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics; Isabelle, P., Charniak, E., Lin, D., Eds.; Association for Computational Linguistics: Philadelphia, Pennsylvania, USA, July 2002; pp. 311–318.
30. Arora, K.; Rangarajan, A. Contrastive Entropy: A New Evaluation Metric for Unnormalized Language Models Available online: https://arxiv.org/abs/1601.00248v2 (accessed on 8 February 2024).
31. Dan Jurafsky; James H. Martin Speech and Language Processing Available online: https://web.stanford.edu/~jurafsky/slp3/ (accessed on 8 February 2024).
32. Li, B.; Han, L. Distance Weighted Cosine Similarity Measure for Text Classification. In Proceedings of the Intelligent Data Engineering and Automated Learning – IDEAL 2013; Yin, H., Tang, K., Gao, Y., Klawonn, F., Lee, M., Weise, T., Li, B., Yao, X., Eds.; Springer Berlin Heidelberg: Berlin, Heidelberg, 2013; pp. 611–618.
33. Sokolova, M.; Japkowicz, N.; Szpakowicz, S. *Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation*; 2006; Vol. Vol. 4304, p. 1021; ISBN 978-3-540-49787-5.