Design Principles and a Software Reference Architecture for Big Data Question Answering Systems

Leonardo Mauro Pereira Moraes[1,2][a], Pedro Calciolari Jardim[1][b] and Cristina Dutra Aguiar[1][c]
[1]*Department of Computer Science, University of São Paulo, São Carlos, Brazil*
[2]*Machine Learning & Artificial Intelligence, Sinch, Stockholm, Sweden*

Keywords: Question Answering, Big Data, Software Reference Architecture, Design Principles.

Abstract: Companies continuously produce several documents containing valuable information for users. However, querying these documents is challenging, mainly because of the heterogeneity and volume of documents available. In this work, we investigate the challenge of developing a Big Data Question Answering system, *i.e.,* a system that provides a unified, reliable, and accurate way to query documents through naturally asked questions. We define a set of design principles and introduce *BigQA*, the first software reference architecture to meet these design principles. The architecture consists of high-level layers and is independent of programming language, technology, querying and answering algorithms. *BigQA* was validated through a pharmaceutical case study managing over 18k documents from Wikipedia articles and FAQ about Coronavirus. The results demonstrated the applicability of *BigQA* to real-world applications. In addition, we conducted 27 experiments on three open-domain datasets and compared the recall results of the well-established BM25, TF-IDF, and Dense Passage Retriever algorithms to find the most appropriate generic querying algorithm. According to the experiments, BM25 provided the highest overall performance.

1 INTRODUCTION

The documents produced by companies have become a valuable source of information for employees and the general public. For example, a product manager may look for information in the technical documentation to certify that a given final product faithfully follows the original project specifications. A decision-making employee may analyze financial reports to positively and quickly react to changes in business conditions. Furthermore, ordinary users can search for details in a product manual to identify if the product fulfills their needs.

However, querying these documents is challenging for users, mainly because of the diversity of the large number of documents available. As a result, the time spent to find a given information can cause a delay in their activities. It is also possible that they access outdated information. Another aspect that burdens the users is obtaining only unhelpful information for their questions. In this paper, we use the terms *query* and *question* interchangeably.

A suitable solution to this challenge is to use a Big Data Question Answering system. The system stores and processes several documents in different formats (e.g. PDF, Open Document Format) and web based files (e.g. files from the Internet). These files are usually spread in knowledge bases like Data Lakes and Document Stores. It also provides a unified and accurate way to query documents. Moreover, the system supports the characteristics of volume, velocity, and variety from the Big Data concept (Laney et al., 2001), thus it is able to scale effectively.

Differently from traditional search engines like Google, the Big Data Question Answering system supports questions in natural language sentences. Furthermore, it understands context, subject, and question intention, among other characteristics (Athira et al., 2013; Zhang et al., 2013; Karpukhin et al., 2020). Therefore, it tends to be more efficient and accurate than traditional search engines (Athira et al., 2013), introducing advantages for many applications. For example, the system can be used to develop a more advanced search engine that scans legal documents or a user-friendly chatbot-based FAQ (Frequently Asked Questions) application that allows

[a] https://orcid.org/0000-0002-9553-9978
[b] https://orcid.org/0000-0001-9475-2526
[c] https://orcid.org/0000-0002-7618-1405

Moraes, L., Jardim, P. and Aguiar, C.
Design Principles and a Software Reference Architecture for Big Data Question Answering Systems.
DOI: 10.5220/0011842700003467
In *Proceedings of the 25th International Conference on Enterprise Information Systems (ICEIS 2023) - Volume 1*, pages 57-67
ISBN: 978-989-758-648-4; ISSN: 2184-4992

users to ask questions about products and services.

The Big Data Question Answering system employs Question Answering (QA) algorithms to query documents. These algorithms are composed of two steps: Document Retriever and Document Reader. The first step takes questions in natural language from the users as input and then looks for related documents that may have the answer to the user questions. The second step produces summarized answers from the retrieved documents.

QA algorithms use Natural Language Processing (NLP) techniques. According to Petroni et al., 2019; Karpukhin et al., 2020; Romualdo et al., 2021, current NLP techniques recall factual knowledge without any fine-tuning, demonstrating their potential as unsupervised open-domain QA algorithms.

However, little attention has been devoted to the problem of investigating *how to build a Big Data Question Answering system*. To face this problem, we must propose design principles and a software reference architecture. Design principles are guidelines, biases, and design considerations that should be followed to select, create, and organize components and features. A reference architecture is a software template where the structures, respective components, and relations provide a concrete system architecture for a particular application or a family of software systems (Galster and Avgeriou, 2011; Klein et al., 2016; Derras et al., 2018).

As discussed in Section 2, there are state-of-the-art solutions in the literature that study the problem *separately*, presenting several drawbacks. On the one hand, general-purpose Big Data architectures do not focus on QA solutions. On the other hand, general-purpose Question Answering architectures do not meet the concepts of Big Data or a software reference architecture. To the best of our knowledge, there is still no solution that considers Big Data and Question Answering architectures in *the same setting* and defines related design principles. In this paper, we investigate this gap in the literature.

We propose *BigQA*, the *first* Big Data Question Answering architecture. The features of the proposed architecture are described as follows. The architecture collects structured, semi-structured, and unstructured data from different sources through the *Input Layer*. Data can be available in several formats, such as multi-documents and web pages. The *Big Data Storage Layer* is needed at the bottom to prepare high-quality data for all kinds of analytical demands required by the upper layers. The *Big Querying Layer* is responsible for processing the users questions that are sent from the *Connection Layer*. Also, all the layers are securely connected by the *Security Layer* and generate analytical data managed by the *Insights Layer*.

We highlight the main contributions of this paper:

1. A set of design principles based on Business (B), Data (D), and Technical (T) requirements to design Big Data Question Answering systems.
2. A software reference architecture to meet the design principles, called *BigQA*.
3. A case study to demonstrate the applicability of *BigQA* to real-world applications.
4. A set of experiments to compare the well-established BM25, TF-IDF, and Dense Passage Retriever algorithms to indicate the best candidate to implement the Document Retriever.

This paper is organized as follows. Section 2 reviews related work. Section 3 introduces the design principles. Section 4 describes the proposed architecture. Section 5 validates the architecture. Section 6 describes experiments performed considering different QA algorithms. Section 7 concludes the paper.

2 RELATED WORK

There are few Big Data and Question Answering architectures that have been proposed in the literature, but they are different from our work on their purpose and features. In Sections 2.1 and 2.2 we survey, respectively, research papers and private technologies regarding these architectures.

2.1 Research Proposals

In this section, we analyze Big Data and Question Answering architectures by dividing them into two groups. Group (i) includes general-purpose software reference architectures developed for data analysis and applications in the context of Big Data. Group (ii) encompasses Question Answering architectures for specific use cases.

In Table 1, we compare the investigated architectures and our work considering the main features of a Big Data Question Answering Architecture discussed in Section 3. We consider the characteristics described as follows.

(c.1) Fits as a software reference architecture.

(c.2) Meets Big Data requirements.

(c.3) Implements security components.

(c.4) Introduces design principles.

(c.5) Implements a Question Answering solution.

(c.6) Can retrieve documents from multiple domains.

Table 1: Characteristics of the proposed *BigQA* architecture and related work.

	Studies	(c.1) Reference Architecture	(c.2) Big Data	(c.3) Security	(c.4) Design Principles	(c.5) QA	(c.6) Open Domain	(c.7) Case Study
Big Data Architecture	(Zhu et al., 2019)	✓	✓		✓			✓
	(Li et al., 2020)	✓	✓		✓			
	(Ataei and Litchfield, 2021)	✓	✓					
	(Cassavia and Masciari, 2021)	✓	✓					✓
	(Yousfi et al., 2021)	✓	✓					✓
Question Answering Architecture	(Sucunuta and Riofrio, 2010)	✓			✓	✓		✓
	(Nielsen et al., 2010)	✓			✓	✓		✓
	(Zhang et al., 2013)					✓	✓	✓
	(Novo-Loures et al., 2020)	✓	✓					✓
	(Karpukhin et al., 2020)					✓	✓	✓
	BigQA (our proposal)	✓	✓	✓	✓	✓	✓	✓

Legend: The ✓symbol indicates the challenges addressed by each study.

(c.7) Evaluates the architecture in a real-world case.

Regarding Group (i), Big Data architectures, the studies detailed in (Zhu et al., 2019; Li et al., 2020; Ataei and Litchfield, 2021; Cassavia and Masciari, 2021; Yousfi et al., 2021) meet Big Data requirements and fit as software reference architectures. However, the proposed architectures are difficult to adapt to the needs of Question Answering due to its constraints and configurations of layers and components. Furthermore, the two studies that introduce design principles consider only technical aspects.

Considering Group (ii), Question Answering architectures, the main objective of the studies described in (Sucunuta and Riofrio, 2010; Nielsen et al., 2010; Zhang et al., 2013; Novo-Loures et al., 2020; Karpukhin et al., 2020) is to propose QA algorithms. Based on these algorithms, these studies develop architectures and apply their solution to real-world case studies. The main drawback of the studies in this group is related to the fact that the proposed architectures are not generic or flexible. The architectures are highly dependent on the proposed algorithms. Specifically assessing the studies of (Sucunuta and Riofrio, 2010; Nielsen et al., 2010), they focus only on technical aspects of the design principles.

The studies described in (Sucunuta and Riofrio, 2010; Novo-Loures et al., 2020) are the closest to our work. Differently from our proposed *BigQA*, the architecture introduced in (Sucunuta and Riofrio, 2010) has a fixed schema for query processing and uses an outdated algorithm compared with current modern NLP algorithms. Also, it does not involve security features or consider Big Data traits. In (Novo-Loures et al., 2020), the authors introduce an NLP architecture based on BDP4J (Big Data Pipelining For Java) to preprocess textual data using data pipelines. This architecture is mainly to perform text processing. Thus, it has not been designed and used to deal with Question Answering. The architecture does not even consider design principles or security artifacts.

Our proposed *BigQA* architecture overcomes the aforementioned shortcomings and fulfills all the characteristics analyzed in Table 1. *BigQA* is a software reference architecture that considers Big Data aspects and the problem of open-domain Question Answering in the same setting. It also includes security aspects to guarantee data protection and to support related general laws. Moreover, *BigQA* consists of high-level layers with specific functionalities. Therefore, it is independent of programming language, QA algorithm, and technology. Besides, our work defines a set of design principles based on business, data, and technical aspects. We also validate *BigQA* by considering a real-world case study.

2.2 Private Technologies

There are some private technologies like IBM Watson Discovery[1], Amazon Kendra[2], and Sinch AskFrank[3] that complies with Big Data Question Answering systems. We do not investigate these technologies in this section because they have a proprietary technology and, to the best of our knowledge, do not provide public research papers.

Furthermore, ChatGPT[4] is a technology recently released by OpenAI. It leverages the information re-

[1]https://www.ibm.com/cloud/watson-discovery

[2]https://aws.amazon.com/pt/kendra/

[3]https://askfrank.ai/home

[4]https://openai.com/blog/chatgpt/

trieval experience, enabling a high level of context understanding and answer generation. ChatGPT has a great capacity of generating natural and human-like answers for complex questions. Therefore, it can answer any query, regardless of its complexity. Apart from the common NLP problems of large models, like hallucination and misleading answers (Ji et al., 2022), the results provided by ChatGPT are very close to human intelligence.

Although these surprising results, we can highlight some drawbacks in the solution:

- Large models are usually trained with static data. Thus, ChatGPT has a frozen knowledge limitation when applied to real-world and dynamic applications.
- It is unknown whether the generated answers are real or completely hallucinated as ChatGPT does not provide references to the source information.
- It is not very clear how one would incorporate *business rules* to ChatGPT reasoning. Therefore, it is difficult to filter out or enrich its answers.

In summary, ChatGPT cannot perform queries that would allow answers based on dynamic data or live events. It also does not incorporate data from external data sources, although data is constantly being updated. Currently, ChatGPT knowledge is based on static Internet data up to 2021. As a result, users can access outdated information or get unhelpful information for their questions. Our proposed *BigQA* architecture overcomes the aforementioned challenges by allowing insertion and updating of data. Furthermore, there are no public research papers that describe and detail ChatGPT logic and implementation.

3 DESIGN PRINCIPLES

Despite the vast literature on Big Data and Question Answering, it remains unclear how to design a suitable Big Data Question Answering system. From our point of view, the system must be designed according to accurate principles that consider *business*, *data*, and *technical* aspects. We highlight that the principles of quality influence the system quality.

In this section, we introduce a set of design principles for Big Data Question Answering systems. These principles are inspired by business models (Müller et al., 2019; Schaffer et al., 2020), the agile manifesto (Misra et al., 2012), and the characteristics of the Big Data concept (Laney et al., 2001). We define each principle as follows.
Principles related to Business (B):

B1: The user must retrieve a proper answer to a given question. The answer may be unknown due to a lack of information in the documents or a misunderstanding of the QA algorithm employed. In this case, the system must inform the user.

B2: The user must access only allowed documents. Therefore, the system should support the implementation of data governance policies to ensure that only authorized users can access portions of data and documents.

B3: The user may write the question using natural language. Thus, the system needs to automatically understand the context, subject, and purpose of the user question.

B4: The answer must summarize the contents of the documents related to the question. The system usually provides the answer in two formats: as a FAQ answer or as intrinsic information within documents, such as parts of texts.

Principles related to Data (D):

D1: The system must persist the documents. Raw documents should be stored in a Data Lake or a similar repository, even if the documents are processed and the system uses a small portion of them. The evolution of NLP algorithms and document processing techniques motivates storing raw documents. Therefore, the system can reuse them in the future if needed.

D2: The system must work with documents from different data sources. Examples of sources include data systems, databases, website crawlers, and web-based collaborative platforms like Confluence[5] and SharePoint[6] pages.

D3: The system must support documents in a variety of formats and with different sizes. Given the characteristics of a data source, a document can be structured according to a specific format, such as web page, PDF, Word, and JSON (JavaScript Object Notation). Furthermore, a document can be small or large, depending on its number of pages. In this case, the system must be able to deal with structured, semi-structured, and unstructured data related to Big Data variety.

D4: The amount of produced documents and texts written can easily reach large volumes of data. Therefore, the system must be able to extend its functionalities to Big Data volume.

D5: After inserting or updating the raw documents, data must be ready for consumption. Therefore,

[5]https://www.atlassian.com/software/confluence
[6]https://www.office.com/sharepoint

the system must be able to process the raw documents considering Big Data velocity. Thus, documents are available at the right time to support the best business decisions.

Principles related to Technical aspects (T):

T1: Modularity. Each component has a specific system functionality and works as an independent module that contains everything necessary to execute its functionality. But, at the same time, all components are connected to deliver the proper value to the system.

T2: Flexibility. The system must easily encompass new components as needed, each with its complexity and functionality. Furthermore, the system should be independent of software programming language and technology.

T3: Analytic. The system must store data, metadata, and usage information for analytical analysis. That is, the system must store analytical information for system managers.

T4: Security. The system must use security artifacts to ensure the system integrity, such as user authentication systems, credentials for adding documents, data governance policies, and encrypted connections between components.

4 PROPOSED ARCHITECTURE

From the design principles introduced in Section 3, we propose *BigQA*, the *first* Big Data Question Answering architecture. It comprises six layers, as depicted in Figure 1.

There are two types of layers: horizontal and vertical. Horizontal layers are those that have an explicit connection between them. For instance, there is a connection between the Communication and the Big Querying layers. Vertical layers refer to those that can connect to any other layer of the architecture. For example, the Security Layer can provide to the Communication Layer secure credentials for an API and to the Insights Layer act as an authentication system for a Data Warehouse.

In this section, we detail the functionality of each layer. We first describe the horizontal layers, *i.e.,* (i) Input; (ii) Big Data Storage; (iii) Big Querying; and (iv) Communication. Then, we discuss the vertical layers, *i.e.,* (v) Security; and (vi) Insights. Finally, we discuss general aspects of the architecture.

Before detailing the layers, we introduce in Example 1 a business application where the *BigQA* architecture is required. We employ this case as a running example throughout this section.

Example 1. Consider a large pharmaceutical company that offers a wide range of health care products and manages many documents, including pharmaceutical leaflets, products reports, financial and contract documents. The company needs a single knowledge base where its employees can quickly look for information using natural language. To comply with this requirement, the company implemented a system using *BigQA* as architecture.

4.1 Input Layer

The Input Layer ingests documents into the system. This layer collects documents from the data sources, such as company files, reports, and trusted websites. It also sends these documents to the Data Lake without (or with minimal) preprocessing. Example 2 illustrates the use of the Input Layer.

Example 2. Data providers use the Input Layer to add documents to the system. In the context of Example 1, data providers are employees who write documents on web-based collaborative platforms and external sources, such as market research documents. These data sources generate different formats of documents, such as Word, JSON, and PDF files.

4.2 Big Data Storage Layer

The first functionality of the Big Data Storage Layer refers to data storage. The layer receives raw documents from the Input Layer and stores them in the Data Lake. For each document, the system performs several related activities. First, it identifies the document type. Then, it applies a set of predefined processing techniques to clean and transform the document data to fulfill the requirements of the Document Store. In the sequence, the system stores the transformed data in this repository. Next, the system updates the Metadata Repository with information about the new, or updated, document obtained from the Data Lake and the Document Store.

The Data Storage Layer is also responsible for data processing. To implement this feature, a big data infrastructure is needed, as well as a distributed and parallel processing framework (*e.g.,* Apache Spark (Zaharia et al., 2010)). The objective is to manage the volume and size of the documents. Besides, the Document Store should store files using a distributed file system (*e.g.,* Hadoop Distributed File System (HDFS) (Shvachko et al., 2010)).

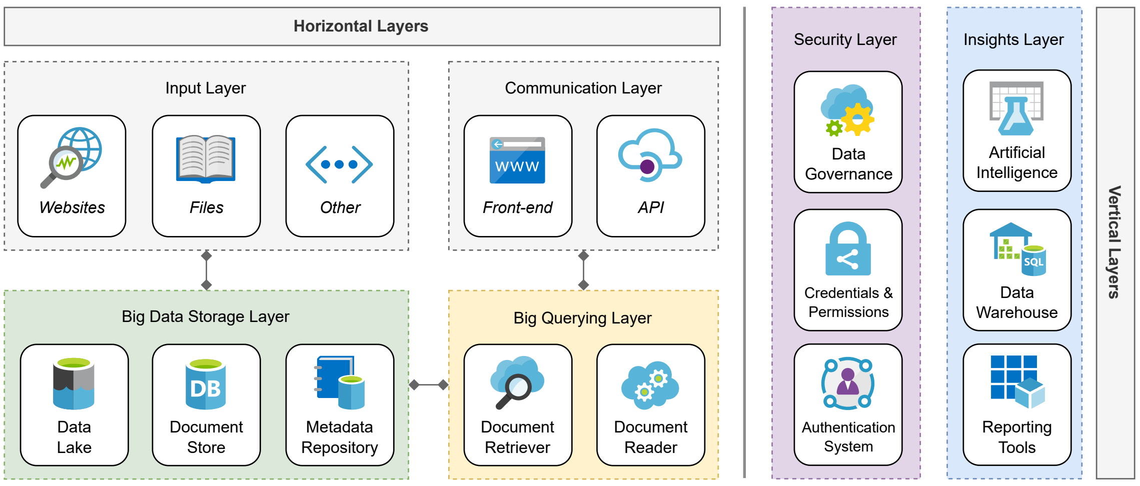

Figure 1: The *BigQA*, the *first* Big Data Question Answering architecture.

4.3 Big Querying Layer

The Big Querying Layer is the core of the architecture. It is the query engine responsible for processing, interpreting, and producing answers to the user questions. This layer retrieves documents from the Big Data Storage Layer, receives queries from the Communication Layer, and produces answers that are sent back to the Communication Layer. Example 3 shows how the Big Querying Answering Layer operates.

Example 3. A pharmacist may need information about the immune system. In the context of Example 1, the pharmacist accesses a web page through the Communication Layer and sends the following question "How do pathogens avoid detection?". The Big Querying Layer processes the query by retrieving the immune system-related documents, summarizing the contents, and producing the answer "pathogens can rapidly evolve and adapt". Finally, the Communication Layer receives the response and presents it to the pharmacist on the web page.

The Big Querying Layer encompasses two components. The Document Retriever retrieves the most valuable documents that may contain the answer to a given question. These documents are obtained from the Document Store using a big data search engine (*e.g.,* Apache Lucene (Lydia et al., 2020)). The second component, Document Reader, examines the retrieved documents and produces a suitable answer for the user. It may execute in parallel on computing clusters (*e.g.,* Kubernetes (Poniszewska-Marańda and Czechowska, 2021)).

4.4 Communication Layer

The Communication Layer acts as an interface through which users submit queries and receive their answers. This layer can encompass several components. In Figure 1, we illustrate two of them. Front-end and API are two components that allow users and applications to send questions to the system.

The Communication Layer requires connectivity through data streaming applications (*e.g.,* Apache Kafka (Lepenioti et al., 2020)). Therefore, it enables near real-time response retrieval.

4.5 Security Layer

The Security Layer addresses security issues like network connection, credentials, and data governance. Because it is a vertical layer, it can apply security artifacts to any other layer. Example 4 illustrates how to use the Security Layer to provide a secure network connection between two components.

Example 4. In implementing the system described in Example 1, the Communication Layer must securely connect to the Big Querying Layer. Otherwise, unauthorized people may access the system and query restricted internal information about the company and its products. The development team must implement an internal network and a firewall system to ensure a secure connection between the components.

The Security Layer consists of, but is not limited to, the components described as follows. The Authentication System is responsible for authenticating users operations. The Credentials & Permissions component should guarantee the definition of appropriate credentials and permissions to apply to the network connection between the components. Finally, Data

Governance refers to managing the availability, usability, integrity, and security of the data in the system according to well-defined policies and constraints.

4.6 Insights Layer

The Insights Layer comprises the data analysis. This layer receives and processes data from multiple layers. Examples of components that are present in the Insights Layer are: (i) Reporting Tools to provide usage reports, data telemetry, and system monitoring; (ii) Data Warehouse, a core component of business intelligence activities that organizes data multidimensionally to support reporting and data analysis; and (iii) Artificial Intelligence models.

4.7 Architecture Discussion

BigQA presents the high-level functionality needed to fit the design principles rather than a restricted set of technology for implementing a Big Data Question Answering system. Therefore, each development team can choose the technology, software programming language, and QA algorithm that better suit their application requirements. Furthermore, each team should adapt the architecture to the application requirements by instantiating only the appropriate layers and components.

BigQA covers the design principles listed in Section 3 as follows: (i) B1-B4: Communication, Big Querying, and Security layers; (ii) D1-D5: Input and Big Data Storage layers; (iii) T1-T2: all layers; (iv) T3: Insights Layer; and (iv) T4: Security Layer.

Regarding the implementation, we encourage lean development using iterative approaches and modular components. Each component has a specific function in the system and can function independently. So, the components can be developed independently and evolutionary, as stated by the Agile manifesto (Misra et al., 2012). To have agile teams, it is essential to simplify the development of the components and to facilitate collaboration across time, location, and organizational boundaries (Schaffer et al., 2020).

5 CASE STUDY

In this section, we present a case study to show how *BigQA* can be deployed to enable a knowledge base containing real-world documents. Our goal is not to perform an extensive analysis of the architecture components. Instead, we implement a real-world case to assess the architecture purpose. Section 5.1 describes how to instantiate *BigQA*. Section 5.2 details queries.

5.1 Architecture Instantiation

Figure 2 depicts the *BigQA* components and layers instantiated in the case study. The Input Layer contains JSON documents obtained from the training sets from two real-world datasets: (i) the Stanford Question Answering Dataset (SQuAD) v1.1 (Rajpurkar et al., 2016); and (ii) the COVID-QA (Möller et al., 2020).

SQuAD is an open-domain Question Answering dataset that stores over 18,800 unique documents with over 87,500 questions and answers about Wikipedia articles. Its contents refers to several different topics, such as pharmacy, antibiotics, databases, software testing, TV series, car companies, and geology.

COVID-QA consists of over 2,000 questions and answers annotated by volunteer biomedical experts on 147 scientific articles related to COVID-19. This dataset is not open-domain. We used it as data augmentation to show that *BigQA* supports adding data from different formats to improve answers.

The Big Data Storage Layer does not keep the raw documents since the datasets were processed before being inserted into the Document Store. Therefore, data transformations converted the JSON documents into Data Storage records. We used the Elasticsearch[7] tool as Document Store. According to Kononenko et al., 2014, "Elasticsearch is an open-source text search engine written in Java that is designed to be distributive, scalable, and near real-time capable".

We employed the Haystack[8] tool to build the Big Querying and Communication layers. Haystack is an open-source framework in Python that supports pipelines for different search applications and includes several state-of-the-art NLP models. We used the well-established QA algorithms BM25 (Robertson and Jones, 1976) and RoBERTa (Liu et al., 2019) as Document Retriever and Document Reader, respectively. Regarding BM25, it was the QA algorithm that provided the best performance in the experiments described in the Section 6.

The code was written in Python using Jupyter Notebooks. It is publicly accessible from the link provided in the Conclusion (Section 7).

5.2 Queries

We describe three queries that can execute on top of the instantiated architecture described in Section 5.1. We issued distinct types of queries to analyze different aspects related to real-world applications. The queries were defined considering the context of the pharmaceutical company detailed in Example 1.

[7]https://www.elastic.co/

[8]https://haystack.deepset.ai/

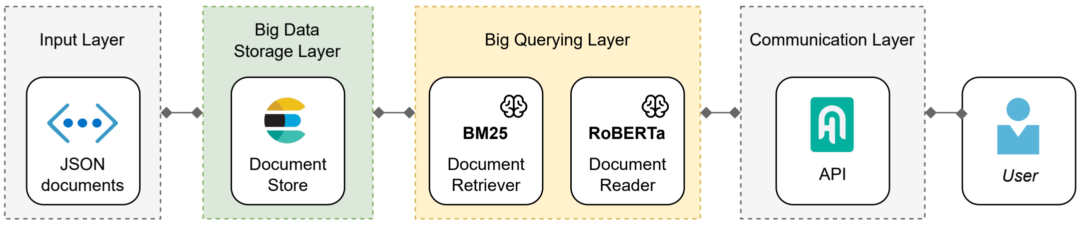

Figure 2: A real-world case study setup using *BigQA*.

For each query, the Document Retriever returned the top-20 documents containing related information, and the Document Reader returned the 3 most likely answers. Therefore, there are three possible answers for each query, each one annotated with a probability score provided by the Data Reader. Higher scores indicate more confidence in the prediction. Each query was an unseen question reformulated from the original dataset to avoid bias from the Document Retriever and Document Reader algorithms. Also, all queries returned answers since we did not evaluate the no-answer scenario.

Query 1: *What law regulates drug marketing in the pharmaceutical industry?* This query represents the interest of pharmacists, marketing, and legal employees in knowing about regulatory laws on drug marketing. It is a named-entity query since it looks for a regulatory law name. This type of query aims at generating an appropriate answer considering that only one document has the correct answer. We executed Query 1 against the SQuAD dataset.

Table 2 shows the results of Query 1. The first two answers refer to documents from the pharmaceutical industry, while the last indicates a United States legal penalty document. The answer with the highest probability score is the right and expected answer. We can conclude that the instantiated architecture can identify the answer to the Query 1 with a score of about 76%.

Table 2: Results of the named-entity Query 1.

Answer	Document	Score
Prescription Drug Marketing Act of 1987	Pharmaceutical industry	76.34%
Food and Drug Administration (FDA)	Pharmaceutical industry	19.77%
Torture Regulation	Capital punishment in the United States	11.01%

Query 2: *When was the Luria–Delbrück?* This query represents the interest of microbiologists in extracting information about a bacterial experiment for antibiotics, which occurred in 1943. It is a date-oriented query since it searches for specific date. This type of query aims at investigating the architecture ability to identify dates from documents. We executed Query 2 against the SQuAD dataset.

Table 3 depicts the results of Query 2. The first document is the only one related to antibiotics; the others are associated with Arnold Schwarzenegger. The answer with the highest probability score is the right and expected answer. However, since the score is below 50%, the Document Reader struggles to consider the answer as correct. Usually, when scores are lower than 50%, the algorithm returns that it was not able to find an answer. We can conclude that the instantiated architecture can identify the answer to the date-oriented Query 2, but the Document Reader should be fine-tuned for date question samples.

Table 3: Results of the date-oriented Query 2.

Answer	Document	Score
1943	Antibiotics	29.89%
14	Arnold Schwarzenegger	6.84%
14	Arnold Schwarzenegger	3.06%

Query 3: *What is the novel Coronavirus?* This query is informative for any pharmaceutical employee and the general public. We executed Query 3 against the SQuAD dataset augmented with the COVID-QA dataset. This type of query, augmented query, explores the architecture ability to extract knowledge from new documents of different formats.

Table 4 depicts the results of Query 3. All the returned documents refer to the Coronavirus and provide a score over 70%. The first and third answers are the correct ones. We can conclude that the instantiated architecture can return the answer to the Query 3 by extracting data from new documents once they are processed and inserted into the Document Store.

5.3 Case Study Discussion

In summary, this case study demonstrated the applicability of *BigQA* to real-world applications, using documents from Wikipedia articles and FAQ ques-

Table 4: Results of the augmented Query 3.

Answer	Document	Score
SARS-CoV-2	COVID-QA	87.70%
Prevention for 2019	COVID-QA	76.78%
SARS-CoV-2	COVID-QA	71.66%

tions and answers. As discussed in Section 5.1, we adapt the instantiated architecture to the application requirements by implementing only the appropriate layers and components. Finally, in Section 5.2, we presented distinct types of queries analyzing different aspects related to business applications. The instantiated architecture was able to properly answer named-entity and data augmented queries, however struggles with the probability of the date-oriented Query 2.

6 DOCUMENT RETRIEVER ALGORITHMS EVALUATION

Because *BigQA* is agnostic, the Document Retriever can implement any algorithm. In this section, we conduct 27 experiments to evaluate three well-established QA algorithms and investigate their recall score. We are motivated by the fact that the use of a higher recall algorithm provides a higher end-to-end querying and answering performance (Karpukhin et al., 2020). Section 6.1 describes the experiments setup and Section 6.2 details the experiment results.

6.1 Experiment Setup

We used almost the same instantiation described in Section 5.1 to evaluate the following well-established QA algorithms: BM25 (Robertson and Jones, 1976), TF-IDF (Sammut and Webb, 2010), and Dense Passage Retriever (DPR) (Karpukhin et al., 2020).

The differences refer to the data characteristics and the employed datasets. We used question-document pairs as input data to carry out this evaluation. The purpose of the experiments was to evaluate the performance of the algorithm to retrieve the correct document for a given question. Moreover, we employed the validation sets of the following three open-domain and real-world datasets:

- SQuAD v1.1 (Rajpurkar et al., 2016), with 10,570 question-document pair samples.
- AdversarialQA (Bartolo et al., 2020), a QA dataset in which humans have created adverse and complex questions, so the models cannot answer these questions easily. This dataset stores 3,000 question-document pair samples.
- DuoRC (Saha et al., 2018), a dataset of movie plot questions and answers on articles from Wikipedia and IMDb, containing 12,845 question-document pair samples.

We compared the performance of the algorithms considering the recall measure. This measure indicates how many times an algorithm retrieves the correct document from the k retrieved documents. We varied the value of k in $[3, 10, 20]$. The literature usually works with $k = 20$ or more. However, to provide fast answers without losing performance, we used these values because business applications require the retrieval of fewer documents.

6.2 Experiment Results

Table 5 depicts the recall results of the investigated algorithms. The results demonstrate that the recall increases as the value of k also increases, indicating that retrieving more documents impacts the probability of recovering the correct document.

In most of the cases, BM25 provided the best performance. This is related to the fact that BM25 extends TF-IDF using a probabilistic information retrieval model, improving recall. Compared to the DPR dense algorithm, BM25 is a sparse algorithm. Dense algorithms are computationally expensive regarding time and secondary memory usage. According to these results, we employed BM25 as the Document Retriever in the case study detailed in Section 5.

Considering the AdversarialQA dataset, DPR provided better results than BM25 and TF-IDF for the values of k equal to 10 and 20. In these cases, DPR was more efficient in understanding the subject and the context of the questions because dense algorithms tend to perform better over complex datasets.

7 CONCLUSION

In this paper, we proposed a set of design principles based on business, data, and technical aspects to support the development of reliable and secure systems. Based on these principles, we introduced *BigQA*, the first Big Data Question Answering architecture. *BigQA* is a software reference architecture composed of the following layers: (i) Input, the ingestion of documents; (ii) Big Data Storage, the storage and processing of the data; (iii) Big Querying, the query engine; (iv) Communication, the user interface; (v) Security, the security artifacts; and (iv) Insights, the data analysis support. The architecture is agnostic, *i.e.,* is independent of programming language, QA algorithm, and technology.

Table 5: Recall results of the QA algorithms investigated to implement the Document Retriever.

	SQuAD			AdversarialQA			DuoRC		
	BM25	TF-IDF	DPR	BM25	TF-IDF	DPR	BM25	TF-IDF	DPR
$k = 3$	**88.74%**	81.12%	69.21%	**71.89%**	69.95%	69.85%	**86.05%**	77.37%	23.29%
$k = 10$	**94.43%**	92.01%	85.72%	81.35%	81.81%	**89.17%**	**91.49%**	87.47%	35.83%
$k = 20$	**96.29%**	95.83%	91.38%	84.89%	85.56%	**99.43%**	**93.76%**	90.80%	44.78%

We validated *BigQA* by implementing a case study in the context of a pharmaceutical company. We used two real-world datasets, one consisting of Wikipedia articles and another storing frequently asked questions about COVID-19. We issued different queries, demonstrating the potential of *BigQA* in the development of real-world applications. We implemented the BM25 algorithm as Document Retriever since it provided the best results according to our evaluation. In this evaluation, we conducted 27 experiments over three open-domain datasets to compare the BM25, TF-IDF, and Dense Passage Retriever algorithms. All code is available on GitHub[9].

We are currently conducting experiments to assess the performance of different algorithms to implement the Document Reader. We also plan to investigate the Insights and Security layers in terms of technologies and algorithms available. Another future work is to analyze new case studies that instantiate the proposed architecture to different real-world applications.

ACKNOWLEDGEMENTS

We thank Sinch, São Paulo Research Foundation (FAPESP), Brazilian Federal Research Agency CNPq, and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, Brazil (CAPES) [Finance Code 001] for support this work. C. D. Aguiar has been supported by the grant #2018/22277-8, FAPESP.

REFERENCES

Ataei, P. and Litchfield, A. (2021). Neomycelia: A software reference architecture for big data systems. In *Proceedings of 28th Asia-Pacific Software Engineering Conference*, pages 452–462.

Athira, P., Sreeja, M., and Reghuraj, P. (2013). Architecture of an ontology-based domain-specific natural language question answering system. *International Journal of Web & Semantic Technology*, 4(4): article number 31.

Bartolo, M., Roberts, A., Welbl, J., Riedel, S., and Stenetorp, P. (2020). Beat the AI: Investigating adversarial human annotation for reading comprehension. *Transactions of the Association for Computational Linguistics*, 8:662–678.

Cassavia, N. and Masciari, E. (2021). Sigma: a scalable high performance big data architecture. In *Proceedings of 29th Euromicro International Conference on Parallel, Distributed and Network-Based Processing*, pages 236–239.

Derras, M., Deruelle, L., Michel Douin, J., Lévy, N., Losavio, F., Pollet, Y., and Reiner, V. (2018). Reference architecture design: A practical approach. In *Proceedings of 13th International Conference on Software Technologies*, pages 633–640.

Galster, M. and Avgeriou, P. (2011). Empirically-grounded reference architectures: A proposal. In *Proceedings of the Joint ACM SIGSOFT Conference - QoSA and Architecting Critical Systems*, page 153–158.

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y., Madotto, A., and Fung, P. (2022). Survey of hallucination in natural language generation. *ACM Computing Surveys*.

Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. (2020). Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Klein, J., Buglak, R., Blockow, D., Wuttke, T., and Cooper, B. (2016). A reference architecture for big data systems in the national security domain. In *Proceedings of IEEE/ACM 2nd International Workshop on Big Data Software Engineering*, pages 51–57.

Kononenko, O., Baysal, O., Holmes, R., and Godfrey, M. W. (2014). Mining modern repositories with elasticsearch. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, page 328–331.

Laney, D. et al. (2001). 3D data management: Controlling data volume, velocity and variety. *META group research note*, 6(70):1.

Lepenioti, K., Bousdekis, A., Apostolou, D., and Mentzas, G. (2020). Prescriptive analytics: Literature review and research challenges. *International Journal of Information Management*, 50:57–70.

Li, Q., Xu, Z., Wei, H., Yu, C., and Wang, S. (2020). General big data architecture and methodology: An analysis focused framework. In *Proceedings of On the Move to Meaningful Internet Systems: OTM 2019 Workshops*, pages 33–43.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov,

[9] *BigQA* experiments and implementation codes. https://github.com/leomaurodesenv/big-qa-architecture

V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Lydia, E. L., Satyanarayan, S., Kumar, K. V., and Ramya, D. (2020). Indexing documents with reliable indexing techniques using Apache Lucene in Hadoop. *International Journal of Intelligent Enterprise*, 7(1-3):203–214.

Misra, S., Kumar, V., Kumar, U., Fantazy, K., and Akhter, M. (2012). Agile software development practices: evolution, principles, and criticisms. *International Journal of Quality & Reliability Management*, 29(9):972–980.

Möller, T., Reina, A., Jayakumar, R., and Pietsch, M. (2020). COVID-QA: A question answering dataset for covid-19. In *Proceedings of the 1st Workshop on NLP for COVID-19 at Association for Computational Linguistics*, page 1.

Müller, M., Vorraber, W., and Slany, W. (2019). Open principles in new business models for information systems. *Journal of Open Innovation: Technology, Market, and Complexity*, 5(6):1–13.

Nielsen, R. D., Masanz, J., Ogren, P., Ward, W., Martin, J. H., Savova, G., and Palmer, M. (2010). An architecture for complex clinical question answering. In *Proceedings of the 1st ACM International Health Informatics Symposium*, page 395–399.

Novo-Loures, M., Pavon, R., Laza, R., Ruano-Ordas, D., and Mendez, J. R. (2020). Using natural language preprocessing architecture (NLPA) for big data text sources. *Hindawi Scientific Programming*, 2020:1–13, article id 2390941.

Petroni, F., Rocktäschel, T., Riedel, S., Lewis, P., Bakhtin, A., Wu, Y., and Miller, A. (2019). Language models as knowledge bases? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2463–2473.

Poniszewska-Marańda, A. and Czechowska, E. (2021). Kubernetes cluster for automating software production environment. *Sensors Jornal*, 21(5): article number 1910.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). SQuAD: 100,000+ Questions for Machine Comprehension of Text. *arXiv e-prints*, page arXiv:1606.05250.

Robertson, S. E. and Jones, K. S. (1976). Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146.

Romualdo, A., Real, L., and Caseli, H. (2021). Measuring brazilian portuguese product titles similarity using embeddings. In *Proceedings of XIII Brazilian Symposium on Information Technology and Human Language*, pages 121–132. SBC.

Saha, A., Aralikatte, R., Khapra, M. M., and Sankaranarayanan, K. (2018). Duorc: Towards complex language understanding with paraphrased reading comprehension. *CoRR*, abs/1804.07927.

Sammut, C. and Webb, G. I., editors (2010). *TF-IDF*, pages 986–987. Springer Science & Business Media.

Schaffer, N., Weking, J., and Stähler, O. (2020). Requirements and design principles for business model tools. In *Proceedings of Americas Conference on Information Systems Proceedings*, pages 1–10.

Shvachko, K., Kuang, H., Radia, S., and Chansler, R. (2010). The Hadoop distributed file system. In *Proceedings of IEEE 26th symposium on mass storage systems and technologies*, pages 1–10.

Sucunuta, M. E. and Riofrio, G. E. (2010). Architecture of a question-answering system for a specific repository of documents. In *Proceedings of 2nd International Conference on Software Technology and Engineering*, pages V2–12–V2–16.

Yousfi, S., Rhanoui, M., and Chiadmi, D. (2021). Towards a generic multimodal architecture for batch and streaming big data integration. *arXiv preprint arXiv:2108.04343*.

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. (2010). Spark: Cluster computing with working sets. In *Proceedings of 2nd USENIX Workshop on Hot Topics in Cloud Computing*, pages 1–7.

Zhang, G., Jiang, T., Bie, R., Liu, X., Wang, Z., and Rao, J. (2013). The architecture of ProMe instant question answering system. In *Proceedings of International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pages 237–242.

Zhu, J. Y., Tang, B., and Li, V. O. (2019). A five-layer architecture for big data processing and analytics. *International Journal of Big Data Intelligence*, 6(1):38–49.