INFORMATION EXTRACTION USING LARGE LANGUAGE MODELS

A MAJOR PROJECT REPORT

Submitted by

DIBYAJYOTI GANGULY [RA2011026010122]
TARUN NEGI [RA2011026010132]

Under the Guidance of

Dr Maivizhi R

(Assistant Professor, Department of Computational Intelligence)

in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING
with specialization in Artificial Intelligence and Machine Learning

DEPARTMENT OF COMPUTATIONAL INTELLIGENCE
COLLEGE OF ENGINEERING ANDTECHNOLOGY
SRMINSTITUTEOFSCIENCEANDTECHNOLOGY
KATTANKULATHUR-603203

MAY 2024

Department of Computational Intelligence
SRM Institute of Science & Technology
Own Work* Declaration Form

This sheet must be filled in (each box ticked to show that the condition has been met). It must be signed and dated along with your student registration number and included with all assignments you submit– work will not be marked unless this is done.

To be completed by the student for all assessments

Degree/Course	: **B.Tech in Computer Science and Engineering with Specialization in Artificial intelligence and Machine Learning**
Student Name	: **Dibyajyoti Ganguly, Tarun Negi**
Registration Number	: **RA2011026010122, RA2011026010132**
Title of Work	: **Information Extraction using Large Language Models**

I / We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism**, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is my / our own except where indicated, and that I/ Wehavemet the following conditions:

- Clearly referenced/listed all sources as appropriate
- Referenced and put in inverted commas all quoted text(from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others(e.g. fellow students, technicians, statisticians, externalsources)
- Compiled with any other plagiarism criteria specified in the Course handbook/University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above. Student1 Signature: Student2 Signature: Date:
If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603 203

BONAFIDE CERTIFICATE

Certified that the project report titled "**INFORMATION EXTRACTION USING LARGE LANGUAGE MODELS**" is the bonafide work of "**DIBYAJYOTI GANGULY (RA2011026010122), TARUN NEGI (RA2011026010132)**" who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Dr Maivizhi R	**Prof. DR. R. ANNIE UTHRA**
SUPERVISOR	**HEAD OF THE DEPARTMENT**
ASSISTANT PROFESSOR	DEPARTMENT OF COMPUTATIONAL INTELLIGENCE
DEPARTMENT OF COMPUTATIONAL INTELLIGENCE	

Examiner I

Examiner II

ACKNOWLEDGMENT

We express our humble gratitude to **Dr.C.Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, **Dr.T.V.Gopal**, for his invaluable support.

We wish to thank **Dr.Revathi Venkataraman**, Professor and Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We are incredibly grateful to our Head of the Department, **Dr.R Annie Uthra**, Professor, Department of Computational Intelligence, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

Our inexpressible respect and thanks to our guide, **Dr.Maivizhi R**, Assistant Professor, Department of Computational Intelligence, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under her mentorship. She provided us with the freedom and support to explore the research topics of our interest. Her passion for solving problems and making a difference in the world has always been inspiring. We also offer our thanks to our Faculty Advisor, **Dr. AL Amutha,** for helping us in our ptoject.

We want to convey our thanks to our Project Panel Members, **Dr.K Babu, Dr. Gopirajan, Dr Kaava Kanagraj,** and Panel Head **Dr.B Hariharan**, Professor, Department of Computational Intelligence, SRM Institute of Science and Technology, for their inputs during the project reviews and support.

We sincerely thank all the staff and students of the Computational Intelligence, Department, School of Computing, S.R.M Institute of Science and Technology, for their help with our project. Finally, we would like to thank our parents, family members, and friends for their unconditional love, constant support, and encouragement.

DIBYAJYOTI GANGULY (RA2011026010122)

TARUN NEGI (RA2011026010132)

ABSTRACT

Efficiently extracting information from documents and/or articles remains a critical challenge in various domains, including academia, industry, and research. As we witness an exponential growth in online sources, the need for efficient and intelligent tools to distill and comprehend this wealth of information becomes more pronounced. The traditional methods of research often fall short in managing the sheer volume of data available, leading to information overload and reduced productivity. In this work, we propose a novel approach leveraging advanced natural language processing techniques for precise information retrieval from PDF documents. Our proposed work leverages the powerful combination of Langchain and OpenAI. This innovative approach is designed to streamline research by allowing users to input articles and extracting pertinent information based on customized prompts. This enables users to provide specific prompts, tailoring the extraction process to focus on particular aspects of the news articles. In this project, we implemented our proposed work, leveraging Langchain, and OpenAI's Language Model (LLM). The experimental results based on accuracy and BLEU score (Billingual Evaluation Understudy) demonstrate the effectiveness of our approach in extracting the information from Portable Document Formats (PDFs).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

AI	Artificial Intelligence
ML	Machine Learning
NLP	Natural Language Processing
PDF	Portable Document Format
LLM	Large Language Models
FAISS	Facebook AI Similarity Search
BANNER	Biomedical Automated Named Entity Recognition
DISCOTEX	Discovery from text extraction
WQCA	Web Query Classification Algorithm
IE	Information Extraction
QA	Question Answering
BLEU	Bilingual Evaluation Understudy

CHAPTER 1

INTRODUCTION

In the dynamic landscape of information dissemination, staying well-informed is crucial, and research plays a pivotal role in understanding and interpreting current events. As there are so many internet news sources available, scholars, journalists, and hobbyists frequently struggle to sort through the deluge of data. By providing a user-friendly platform where users may input documents containing news, stories and return focused and concise summaries based on their defined prompts, we seek to streamline this process. In addition to extracting pertinent details, we intend to condense articles' substance to give users a thorough rundown of the material.

In today's information-rich environment, the ability to efficient extraction of relevant responses from massive stores of textual data is essential. Information Retrieval (IR) system finds the relevant documents from a large dataset according to the user query [1]. The need for advanced tools that can swiftly and accurately search through digital documents to locate specific information that users are seeking for is growing, as these documents, especially those in the Portable Document Format (PDF), continue to increase at an exponential rate.

Information extraction systems mostly serve the purpose of unique data processing modules targeting the retrieval of information as required by the user [3]. It is a technique that can be used to obtain information from documents, either using knowledge engineering approach or automatic training approach [4]. Traditional methods of extracting data from PDF documents usually include manual scanning or keyword searches, which can be take a lot of time and be imprecise, when it involves big or complex documents. Furthermore, users must sift through possibly unnecessary material in order to find the precise solutions they need, because these techniques typically return full papers or portions of them.

The suggested model-driven solution, on the other hand, seeks to expedite this procedure by allowing users to ask natural language queries (prompts). The most difficult aspect of using a Question Answering System is bridging the gap between the questions and the corpus to discover relevant replies. The questions and corpus are in distinct contexts; to locate the answers, the inference if one candidate sentence meets the user's request is required [5]. The pre-requisite knowledge needed to understand this study is listed below.

1.1 Artificial Intelligence

Artificial intelligence (AI) is the creation and application of computer systems and software designed to do tasks that normally require human intelligence. These tasks may involve problem solving, learning from experience, comprehending natural language, identifying patterns, making judgments, and others. Its ideal feature is its ability to rationalize and act in order to attain a specified goal. AI research began in the 1950s, and in the 1960s, the United States Department of Defense trained computers to mimic human reasoning.

Artificial intelligence can be used in a variety of industries, including healthcare, to recommend medicine dosages, find treatments, and assist with surgical procedures in the operating room. Other examples of AI-powered technologies are chess computers and self-driving autos.

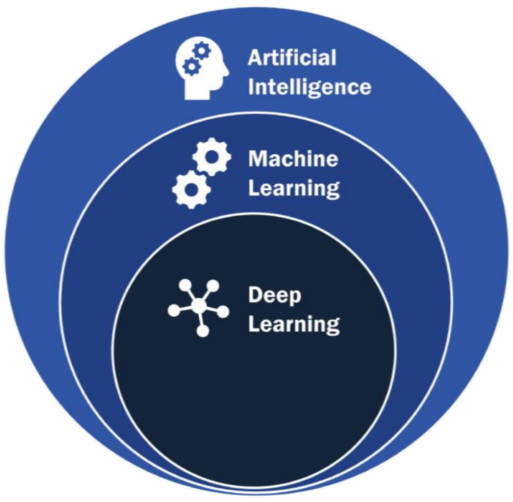


Fig 1.1 Artifical Intelligence

In Figure 1.1, artificial intelligence is depicted along with its subsets i.e. machine learning and deep learning. AI makes use of a variety of techniques and technologies, including Machine Learning (ML), which allows systems to improve their performance based on data without having to be explicitly programmed, and Deep Learning, a more advanced subset of ML that processes complex information such as images and text using intricate neural networks.

AI has a wide range of applications. Natural Language Processing (NLP) enables computers to interpret and respond to human language, resulting in chatbots, virtual assistants, and language translation services. Another area of AI is computer vision, which includes interpreting visual input and has applications such as facial recognition, object detection, and medical imaging analysis. Robotics and automation rely on artificial intelligence to manage physical tasks and control robotic equipment. Meanwhile, Expert Systems imitate human specialists' decision-making processes in specialized disciplines, providing significant insights into businesses such as healthcare and finance.

1.2 Machine Learning

Machine Learning (ML) is a subfield of artificial intelligence that involves creating algorithms and statistical models that allow computers to learn from data and improve their performance over time without being explicitly programmed. This learning process is based on spotting patterns, making predictions, and adjusting to new information, which enables machines to do complicated tasks with greater precision and efficiency.

The core idea behind machine learning is that systems can automatically learn from experience. This expertise usually takes the form of big datasets that are evaluated by algorithms to extract insights and construct predictive models. These models can then be applied to a wide range of tasks, including data classification, recommendation making, picture recognition, natural language understanding, and more. Deep learning and machine learning differ in how their algorithms learn. Labeled datasets, also known as supervised learning, can be used to inform "deep" machine learning algorithms, however they are not always necessary.

The deep learning technique may accept unstructured data in its raw form (e.g., text or photos) and automatically identify the collection of characteristics that separate distinct types of data from one another.

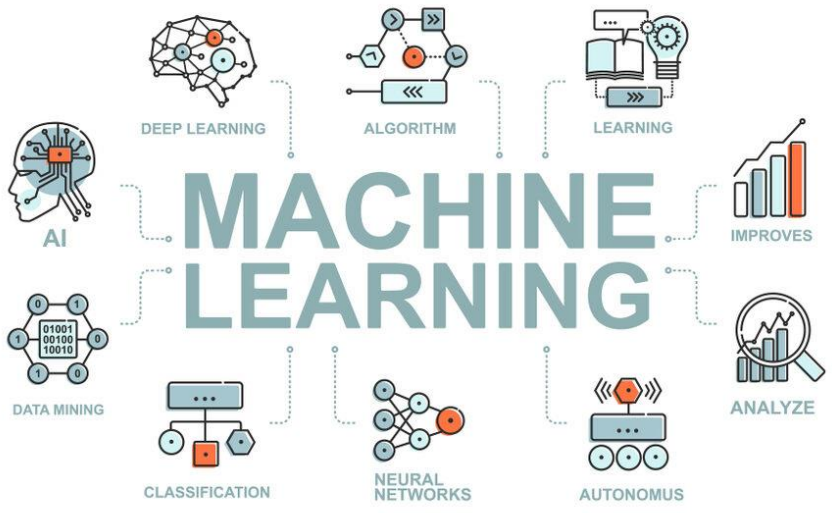


Fig 1.2 Machine Learning

It is depicted in Figure 1.2 that machine learning is a technology that interconnects all the other technologies. Supervised learning, a subset of machine learning, often known as supervised machine learning, is defined as the use of labeled datasets to train algorithms to correctly classify data or predict outcomes. As input data is entered into the model, the weights are adjusted until the model is properly fitted. This occurs as part of the cross validation procedure to prevent the model from overfitting or underfitting. Unsupervised learning, also known as unsupervised machine learning, is the analysis and clustering of unlabeled datasets using machine learning algorithms. These algorithms identify hidden patterns or data groups without the need for human intervention.

The practical applications of machine learning are numerous and expanding. Machine learning is used in business to run recommendation engines, analyze customer behavior, and detect fraud. It is used in healthcare to aid in disease diagnosis and individualized treatment regimens. It promotes technological improvements in natural language processing, self-driving cars, and computer vision.

1.3 Natural Language Processing (NLP)

Natural language processing is the ability of a computer program to understand human language as it's spoken and written -- referred to as natural language. It's a component of artificial intelligence (AI). NLP employs a wide range of methods to give computers the same level of natural language comprehension as people. Natural language processing uses artificial intelligence (AI) to analyze and interpret spoken or written language in a way that is understandable to a computer.

The two primary stages of natural language processing are the creation of algorithms and data preprocessing. Text data is cleaned and prepared during the data preprocessing stage so that machines can analyze it more easily. Tokenization is one technique used to replace sensitive data with nonsensitive tokens, while stop word removal is another technique used to remove frequent words and leave behind unique words that contain important information. Words with inflections are grouped together for processing by lemmatization and stemming, which reduce them to their basic forms.

Part-of-speech tagging classifies words according to their grammatical functions. Algorithms are designed to process the preprocessed data. Rule-based systems, which depend on meticulously crafted linguistic rules, and machine learning-based systems, which make use of statistical techniques and modify their methods in response to training data.

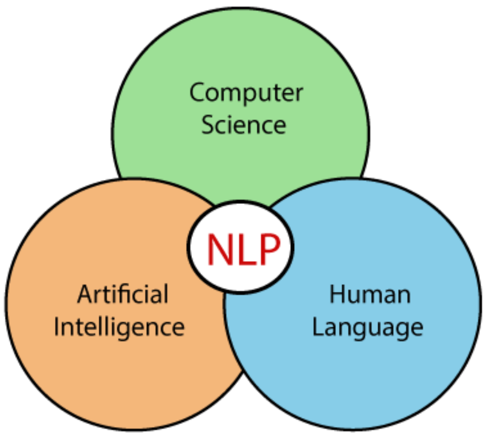


Fig 1.3 Natural Language Processing

As we can observe in Figure 1.3, NLP is an intersection of Computer Science, Artifical Intelligence and Human Language. Analysis paralysis occurs when the sheer volume of knowledge becomes overwhelming, impeding the ability to make informed decisions or formulate specific research questions. Furthermore, the quality of information itself may be jeopardized. In the rush to report the newest news or garner internet attention, truth and nuance can frequently be overlooked. Natural Language Processing approaches provide a ray of hope, allowing us to navigate this digital labyrinth more efficiently and precisely. Using the capabilities of NLP, we can create sophisticated information retrieval systems that understand the meaning and context of online content, rather than merely the existence of keywords.

1.4 Large Language Models (LLM’s)

Large Language Models (LLMs) are artificial intelligence systems that understand, synthesize, and manipulate human language on a large scale. These models are based on deep learning architectures, specifically transformer networks, which enable them to analyze massive volumes of text input while learning complicated linguistic patterns. LLMs are trained on massive datasets, which are frequently derived from the internet, resulting in a comprehensive comprehension of language, context, and even particular knowledge about a wide range of topics.

Large Language Models (LLMs) are artificial intelligence systems that understand, synthesize, and manipulate human language on a large scale. These models are based on deep learning architectures, specifically transformer networks, which enable them to analyze massive volumes of text input while learning complicated linguistic patterns. LLMs are trained on massive datasets, which are frequently derived from the internet, resulting in a comprehensive comprehension of language, context, and even particular knowledge about a wide range of topics. LLMs form the foundation of numerous natural language processing (NLP) applications, including chatbots, virtual assistants, language translation services, and automated content generating tools. They can respond to text messages in a human-like manner, hold meaningful discussions, summarize lengthy documents, and answer questions using the knowledge they gained throughout training.

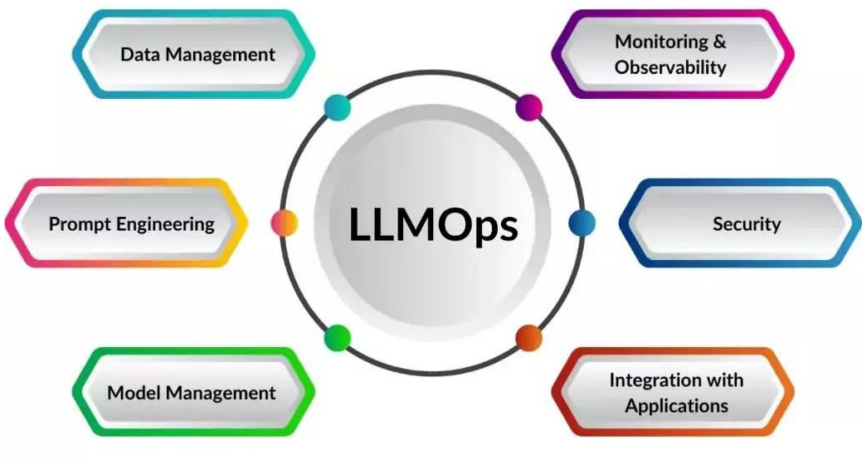


Fig 1.4 Large Language Model

In Figure 1.4, the applications of large language models in different fields is depicted i.e. data management, prompt engineering, security, model management, integration and moniroting. Despite their amazing powers, LLMs also face many challenges. As they are trained on big datasets from various sources, they may unwittingly develop and perpetuate biases, resulting in potentially harmful or discriminating effects. Furthermore, because they do not intrinsically grasp the content they receive, they might produce convincing but inaccurate information, sometimes known as "hallucinations." This raises problems about dependability and trustworthiness.

Overall, LLMs offer a substantial advancement in AI's ability to engage with human language, paving the way for a variety of applications and services. However, their complexity and the risks associated with prejudice and misinformation necessitate rigorous management to ensure ethical and responsible implementation in real-world circumstances.

1.5 Langchain

Langchain, a powerful and adaptable framework, acts as the information architect, methodically deconstructing the intricate structure of PDF documents. It maps the PDF document's layout, accurately identifying sections, paragraphs, and tables. It goes beyond the surface layer of text to recognize headers, footers, and other structural components, resulting in a thorough grasp of the document's organization. This analysis enables browsing the page with pinpoint accuracy, ensuring that no important information is overlooked. LangChain is an open-source framework for developing applications using large language models (LLMs). LLMs are huge deep-learning models that have been pre-trained on vast quantities of data and can respond to user inquiries, such as answering questions or making graphics based on text prompts.

Langchain streamlines intermediate processes in the development of such data-responsive programs, allowing for more efficient quick engineering. It is intended to make it easier to construct a variety of language-powered applications, such as chatbots, question-answering, content production, summarizers, and so on. It also enables enterprises to repurpose LLMs for domain-specific applications without requiring retraining or fine-tuning.

It simplifies artificial intelligence (AI) development by abstracting the complexities of data source integrations and allowing for quick refinement. Developers can use custom sequences to swiftly develop complicated applications. To save time on development, software teams can alter the templates and libraries rather than developing business logic. It provides AI developers with tools for connecting language models to other data sources. It is open-source and backed by a thriving community.

This framework also provides tools and modules for constructing and altering existing chains, allowing developers to design sophisticated applications. Agents, a special form of chain, use the large langauge model to identify the best response sequence for a query, for information processing, storage, retrieval, and searching. This improves the replies by allowing developers to generate semantic information representations from word embeddings stored in databases. Memory capabilities are also enabled, which allows incorporation of previous interactions into the system's answers.

1.6 Integration with Multiple Technologies and Scope

Many unique and innovative ideas have been implemented in this particular work. One of them is semantic understanding, where the system employs advanced natural language processing techniques like semantic similarity search and vector embeddings to comprehend and analyze the content of PDF documents. This goes beyond simple keyword searches and manual scanning, allowing for deeper understanding and more accurate retrieval of information. Also, The use of metrics such as BLEU scores for comparing the performance of different language models adds a quantitative dimension to the evaluation process. This allows for objective assessment of the system's performance and helps identify areas for improvement. Traditional information retrieval systems often rely on a single technology, such as keyword matching or basic natural language processing (NLP) techniques. Whereas, our approach helps in identifying the most relevant information within the PDF and answer user queries with greater precision. It can understand the context and meaning of the text, allowing it to answer complex questions and even identify subtle nuances within the PDF document. It allows for more targeted and personalized information retrieval from news articles.

The tool finds application in various parts of our life as well. Journalists and reporters rely on timely and accurate information to craft compelling stories and stay ahead in the competitive news industry. It streamlines the research phase by enabling them to quickly extract key information from multiple news articles. This aids in fact-checking, background research, and staying updated on developing stories. By efficiently gathering relevant data from various sources, journalists can produce high-quality content more efficiently, enhancing their productivity and ensuring the accuracy and comprehensiveness of their reporting.

Largescale businesses operate in dynamic environments where staying informed about industry trends, market developments, and competitor activities is crucial for strategic decision-making. The PDF query tool serves as a valuable asset for competitive intelligence, allowing businesses to extract and summarize information from news articles relevant to their industry and competitors.

By monitoring news coverage, businesses can identify emerging trends, assess market sentiment, and gain insights into competitor strategies. This enables them to adapt their own strategies, seize opportunities, and mitigate risks effectively, ultimately enhancing their competitive advantage in the marketplace.

Marketing professionals can leverage the tool to extract consumer insights, market trends, and competitor strategies from news articles and industry reports. This informs marketing campaigns, product development strategies, and market positioning efforts, driving business growth and market success. Researchers can utilize the tool to quickly extract key findings and data from vast collections of academic papers. Imagine a literature review process streamlined by automatically identifying relevant statistics and summarizing research arguments within numerous PDFs. This newfound efficiency empowers researchers to delve deeper into specific topics and accelerate their progress. Students grappling with complex textbooks or academic articles can leverage the tool to clarify concepts and identify key points.

Overall, our proposed approach intends to advance the field of question-answering systems by offering a novel approach specifically tailored for precise and accurate response extraction from PDF documents. It enable users to efficiently retrieve the vast amount of information included in PDFs by combining advanced natural language processing techniques with meticulous document preparation methods. We aim to address the difficulties presented by the intricate structure of PDF documents by utilizing the most recent advancements in natural language processing. It aims to close the gap between the vast amount of information included in PDF documents and the user's requirement for prompt and precise responses, and to fully realize the potential of PDFs as insightful and useful sources of knowledge by fusing state-of-the-art natural language processing algorithms with meticulous document preparation approaches.

1.7 Objectives

The proposed approach enables users to efficiently retrieve the vast amount of information included in PDFs by combining advanced natural language processing techniques with meticulous document preparation methods. The major objectives of this thesis are:

- To develop a tool that accurately extracts key information from articles to ensure the reliability of the xtracted content and enhancing the system's ability to understand and interpret the context of articles, addressing challenges related to ambiguous language, sarcasm, and nuanced expressions.
- To provides users with the ability to input customized prompts, tailoring the information extraction process to meet individual or specific organizational needs and improving the adaptability of the system across diverse domains, ensuring consistent performance regardless of the topic of the articles.
- To offer the ability to generate concise summaries of the PDF, providing users with a quick overview of the key points.

1.8 Organization of the report

Chapter 2 includes a compilation of the works related to our approach, describing their methodologies, advantages and disadvantages.

Chapter 3 briefs on the proposed architecture of our approach, briefing on the role of each component.

Chapter 4 consists of a detailed description of the proposed methodology and the various steps involved in our approach.

Chapter 5 comprises the results and discussions of our proposed approach.

Chapter 6 lists the conclusions drawn for the methodology conducted, as well as briefing on the future enhancements of the proposed work.

CHAPTER 2

LITERATURE REVIEW

In recent years, the field of question-answering with PDF document inputs has garnered significant attention due to the ever-expanding volume of digital documents and the growing demand for efficient information retrieval systems. Traditional methods for accessing information within PDF documents, such as keyword searches and manual scanning, often prove inadequate when dealing with extensive or complex documents, leading to inefficiencies and inaccuracies in information retrieval. Recognizing this challenge, researchers have proposed innovative approaches aimed at enhancing the extraction and understanding of information from PDFs. This chapter provides insights into the diverse methodologies and advancements made in this domain, ranging from novel approaches for PDF document processing and layout analysis to the integration of multi-modal techniques for visually-rich document understanding. The papers have been classified on the basis of two unique factors i.e. automatic text etraction and question-answering systems.

2.1 Automatic Text Extraction

Raymond J.Mooney et al. [2] recognized the challenge that while unstructured text contains valuable information, it is difficult to analyze and use effectively without converting it into a structured format. They proposed an approach that combines text mining techniques with information extraction (IE) methods. Information extraction involves identifying specific pieces of information (such as names, dates, events, etc.) from text and structuring them into a usable format. This enables the conversion of unstructured text data into structured formats, making it easier to analyze and use for various applications. However, this approach has problems when a novel extracted entity is represented by similar but not identical strings in different documents.

Imran Rasheed et al. [6] have proposed a novel approach in association with the need of constantly upgrading the information retrieval system to face the difficulties as the user inquiries become more complicated with time. The system involves standard preprocessing steps like text normalization and stop-word removal for Urdu text. The retrieval process involves techniques like ranking documents based on their similarity to the expanded query. By expanding queries with relevant terms, the system aims to capture the user’s intent more comprehensively but, there’s a risk of expanding queries with irrelevant terms, potentially leading to retrieving less relevant documents

Rasika P. Saste et al. [7] have used Stanford dependency grammar to provide an accessible description of the grammatical links in a sentence. This paper discusses an information extraction approach in which extraction requirements are represented in the form of database queries. This effort tries to reduce processing time when installing an upgraded component compared to a standard technique.

Although they have certain shortcomings, information extraction (IE) and knowledge discovery in databases (KDD) are both helpful methods for finding information in textual corpora. Information extraction can find pertinent textual subsequences, but it typically misses fresh information that has come to light and patterns in a text, making it unable to create new facts or theories. Information extraction has shortcomings that could be addressed by new data mining techniques and methodologies, which work in tandem with it. In this paper, Christina Feilmayr et al. [8] attempts to integrate data mining and information extraction methodologies.

Ferrucci et al. [10] propose UIMA (Unstructured Information Management Architecture) as a framework for handling unstructured information processing tasks, particularly relevant in corporate research environments. UIMA’s modularity enables building complex information processing workflows by combining various components. This promotes code reusability and simplifies maintenance. It also facilitates the integration of diverse tools and technologies within a single framework, promoting interoperability and streamlining workflows. However, UIMA’s modularity can introduce complexity, particularly for simpler information processing tasks.

Cunningham et al. [11] presented GATE (General Architecture for Text Engineering), a framework and graphical development environment for Natural Language Processing (NLP) tasks. GATE offers a robust platform for users to develop and deploy NLP tools and resources. The architecture facilitates not only building custom NLP applications (like information extraction) but also corpus construction, annotation, and application evaluation. GATE's support for Unicode allows for developing applications that handle multilingual text data.

Chen et al. [12] addressed the challenge of optimizing complex information extraction programs designed to handle evolving text data. Their approach focuses on identifying stable regions within the text data that are less prone to change. By selectively re-extracting information only from these stable regions and leveraging pre-computed results for unchanged parts, they aim to improve efficiency. This strategy offers advantages like reduced processing time and resource consumption, particularly beneficial for dealing with large and frequently updated text collections.

In the field of information extraction, Sarawagi et al. [13] surveyed techniques for automatically extracting structured information like entities, relationships, and attributes from various unstructured sources. The paper explores methods for optimizing different stages of the information extraction pipeline, including adapting to evolving data sources, integrating extracted information with existing knowledge bases, and handling uncertainties in the extraction process. This survey provides a valuable resource for researchers and developers interested in the fundamental concepts of information extraction technologies and serves as a reference for those designing or implementing information extraction models.

Leaman et al. [14] introduced BANNER (Biomedical Automated Named Entity Recognition) as a software system that functions as a survey and evaluation tool for named entity recognition (NER) techniques in the biomedical domain. BANNER operates by processing text in stages: first, it extracts text from PDF documents. Then, it preprocesses the text by breaking it into smaller chunks. To enhance semantic understanding, language embeddings are downloaded from external sources like OpenAI. Next, the text segments are indexed using the FAISS library from Facebook AI.

Cafarella et al. [15] presented a search engine designed specifically for use within natural language applications. Their approach focuses on indexing smaller, manageable chunks of text extracted from web documents. These text segments are indexed using aframework called FAISS (Facebook AI Similarity Search) to enable efficient retrieval based on semantic similarity. When a natural language application submits a query, the system searches the indexed text segments to find relevant passages. This architecture allows natural language applications to directly access and utilize information from the web without requiring complex web scraping techniques.

Cheng et al. [16] presented an approach to building an "Entity Search Engine" that facilitates agile information integration from the web. Their system focuses on "best-effort" information retrieval, aiming to deliver useful results even with incomplete data. The methodology involves text extraction from PDF documents, followed by preprocessing and segmentation using techniques like langchain This architecture enables the system to handle diverse web data and potentially provide relevant results.

Agichtein et al. [17] presented a method for querying text databases to facilitate efficient information extraction. Their approach revolves around transforming unstructured text documents into a structured format suitable for retrieval. This involves splitting the text into manageable segments and then indexing these segments using an indexing framework like FAISS (Facebook AI Similarity Search). User queries are then matched against the This technique offers efficient information extraction by leveraging established indexing structures for text retrieval. However, the process requires pre-processing steps like text segmentation and indexing, which might add overhead to the system.

Kim et al. [21] presented an unsupervised approach for extracting domain-specific terms from text documents. Their method leverages co-occurrence statistics to identify terms that frequently appear together and are likely indicative of a specific domain. This unsupervised approach avoids the need for manually labeled training data, which can be a bottleneck in supervised learning techniques. The approach prioritizes terms with high mutual information, suggesting a strong relationship between the terms within the domain context. This method struggles to capture nuanced domain-specific terminology.

2.2 Question Answering Systems

Naw Thiri Wai Khin et al. [1] have proposed the Web Query Classification Algorithm through the use of graph databases in NoSQL. The web inquiries are categorized by this system according to each feature and each pre-established target category. The input query is initially categorized into online search taxonomies (characteristics) in the web query classification process. Domain phrases are then taken out of the query and categorized into the appropriate groups that are kept in the NoSQL database. Basic elements of the IR include document indexing, searching, and ranking. The standard interface of the existing information retrieval systems, such as Web search engines, consists of a single keyword-accepting input box.

Song Liu et al. [5] have suggested that the biggest challenge in question-answering systems is navigating the distance between the queries and the corpus to identify the relevant responses. The corpus and the questions are in distinct contexts, thus it is important to deduce whether a particular candidate sentence satisfies the user's request in order to determine the responses. Inference regarding the user and the utterances is the focus of pragmatic thinking. It serves as a bridge between the corpus and the queries. It should be noted that analyzing paragraph acts and incorporating pragmatic information adds complexity to the information extraction process, requiring sophisticated natural language processing techniques.

In 2006, W. Yue, Z. Chen, and X. Lu [18] presented a ground-breaking method for classifying and expanding queries to retrieve information. The methodology is inspired by the observation that, even with excellent recall, very short searches using standard information retrieval techniques sometimes have poor precision. Their method used text classification and query expansion to try and gather more relevant documents. The trials' findings demonstrated that the suggested algorithm outperforms the conventional query expansion techniques in terms of efficiency and precision.

CHAPTER 3

SYSTEM ARCHITECTURE

The system architecture and design of the information extraction tool, as observed in Figure 3.1, are meticulously crafted to optimize the process of information retrieval and analysis from PDF documents. At its core, the architecture harnesses the synergistic capabilities of Langchain and OpenAI's Language Model (LLM), ensuring seamless integration and efficient operation.

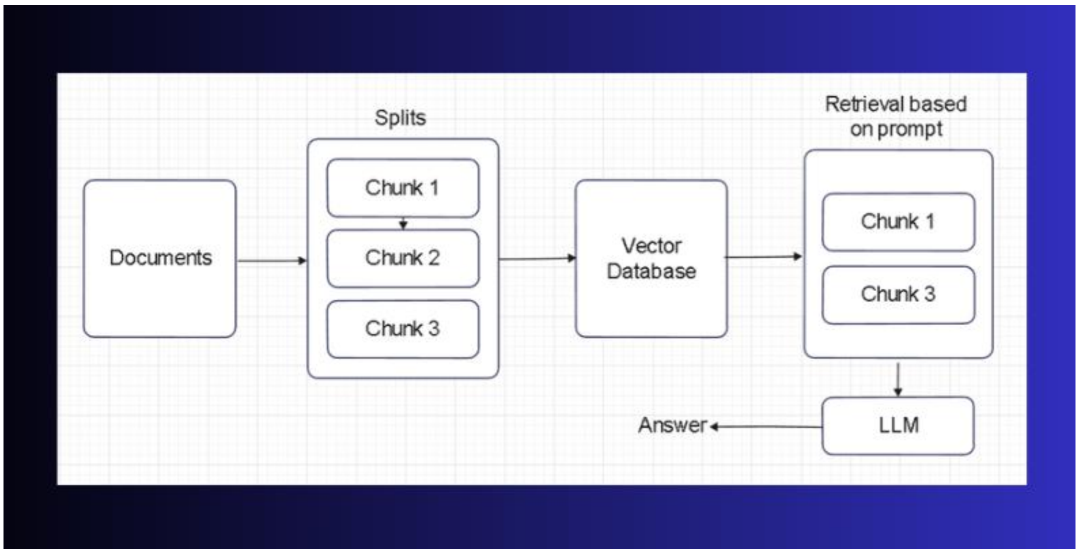


Fig 3.1 Schematic diagram of system architecture

3.1 Chunk Splitting

This component serves as the initial gateway for PDF documents, enabling efficient access and extraction of textual data. It employs advanced parsing techniques and Langchain's text processing capabilities to handle diverse document formats and structures. It manages document metadata, annotations, and auxiliary information to enrich the extracted content and provide context for downstream processing. It ensures accurate extraction of content, enhancing the scope of documents that can be processed.It serves as the initial gateway for PDF documents, enabling efficient access and extraction of textual data. Leveraging advanced parsing techniques and Langchain's text processing capabilities, the Text Extraction Module ensures seamless integration and operation within the PDF-query tool. Its robust framework allows for the handling of diverse document formats and structures, ranging from standard text-based PDFs to more complex layouts and scanned documents. By employing sophisticated algorithms, the module accurately extracts content while preserving the original format.

It goes beyond simple text extraction by incorporating mechanisms for managing document metadata, annotations, and auxiliary information. This comprehensive approach enriches the extracted content, providing additional context for downstream processing and analysis. Metadata such as author information, creation date, and document versioning contribute to a more holistic understanding of the document, while annotations and supplementary data offer insights into the document's content hierarchy and context. Additionally, it prioritizes accuracy and efficiency in content extraction, ensuring that even complex documents are processed with precision.

Langchain's cutting-edge technology enhances the module's parsing capabilities, enabling it to navigate through intricate document structures with ease. By understanding the nuances of language and document layout, the module can extract textual data comprehensively, regardless of the document's complexity or language. It serves as the foundation for information retrieval and analysis within the PDF-query tool, providing users with a reliable and efficient means of accessing valuable insights from PDF documents. Its seamless integration with Langchain and advanced text processing capabilities empower users to extract meaningful content from a wide range of document sources, facilitating informed decision-making and knowledge discovery.

3.2 Semantic Analysis

It plays a pivotal role in understanding the semantics and context of extracted text. It converts textual data into numerical vectors (embeddings) to encode semantic meaning, enabling the system to discern relationships and identify key concepts effectively. Utilizing advanced NLP techniques, it analyzes both syntactic and semantic structures, enhancing the system's comprehension of complex information. Furthermore, it integrates domain-specific knowledge and ontologies to enrich the semanticrepresentation, facilitating more accurate analysis and interpretation of domain-specific documents.

At the heart of the PDF-query tool, the Semantic Analysis Module serves as the backbone for understanding the underlying semantics and context of the extracted textual data. It employs sophisticated natural language processing (NLP) techniques to convert textual information into numerical vectors, known as embeddings. These embeddings encode the semantic meaning of the text, allowing the system to discern intricate relationships between concepts and entities effectively. Leveraging advanced NLP algorithms, the Semantic Analysis Module analyzes both syntactic and semantic structures within the text, enabling a comprehensive understanding of the content's meaning. By capturing subtle nuances in language, such as context, ambiguity, and connotation, the module enhances the system's ability to comprehend complex information and facilitate precise information retrieval.

It integrates domain-specific knowledge and ontologies to enrich the semantic representation of the text. By incorporating specialized knowledge repositories and domain-specific ontologies, the module gains a deeper understanding of the subject matter, thereby improving the accuracy and relevance of its analyses. This integration not only enhances the semantic understanding of the text but also enables the module to contextualize information within the domain's specific knowledge framework. In addition to its semantic analysis capabilities,it utilizes the FAISS (Facebook AI Similarity Search) library for efficient similarity search and retrieval of embeddings. FAISS is a powerful library for similarity search and clustering of dense vectors, optimized for large-scale applications. By leveraging FAISS, the module can quickly retrieve similar embeddings, enabling fast and efficient information retrieval from the vector database.

Hence, it plays a crucial role in unlocking the latent semantic meaning embedded within extracted text. Its combination of advanced NLP techniques, integration of domain-specific knowledge, and utilization of FAISS for efficient similarity search make it a cornerstone of the PDF-query tool, facilitating nuanced semantic analysis and empowering users to derive actionable insights effectively from PDF documents.

3.3 Question Answering

At the core of the system, it processes user queries and retrieves relevant information from indexed text segments. It employs a combination of NLU and IR techniques to generate context-aware responses tailored to the user's intent. By analyzing semantic similarity between user queries and indexed text segments, it identifies relevant passages and extracts concise answers, ensuring a seamless and intuitive user experience. The cornerstone of the PDF-query tool's functionality, the Question-Answering Module serves as the primary interface for users to interact with the system and retrieve relevant information from indexed text segments. This module is designed to process user queries with precision and efficiency, leveraging a combination of Natural Language Understanding (NLU) and Information Retrieval (IR) techniques.

When a user submits a query, it first employs NLU techniques to understand the intent and context behind the query. By analyzing the semantic similarity between the user's query and the indexed text segments, the module identifies relevant passages and extracts concise answers tailored to the user's intent. This context-aware approach ensures that users receive accurate and relevant information in response to their queries, enhancing the overall user experience. Furthermore, it incorporates mechanisms for handling ambiguity, context switching, and multi-turn dialogue. Ambiguity resolution techniques enable the module to disambiguate between multiple interpretations of a query, ensuring that the retrieved information aligns with the user's intended meaning. Additionally, the module seamlessly manages context switching, allowing users to refine their queries or switch topics during a single interaction session without losing context.

In scenarios where a user engages in multi-turn dialogue, it maintains context across interactions, enabling a coherent and fluid conversation flow. This capability enhances user engagement and satisfaction by facilitating meaningful interactions with the system over multiple turns of conversation. It continuously learns and adapts to user feedback, refining its understanding of user intent and response generation mechanisms over time. Real-time feedback mechanisms enable the module to adapt to evolving user needs and preferences, ensuring that the quality and relevance of responses provided continually improve. Also, it represents a sophisticated blend of NLU and IR techniques, tailored to provide users with a seamless and intuitive experience when retrieving information from

PDF documents. Its robust performance in handling ambiguity, context switching, and multi-turn dialogue, coupled with its adaptability to user feedback, makes it a vital component of the tool, empowering users to extract actionable insights effectively. These integrations empower the system with unparalleled linguistic analysis and extraction capabilities, facilitating efficient information retrieval and analysis from PDF documents.

3.4 LLM Usage

Langchain, as a cornerstone technology integrated into the text extraction module, revolutionizes the process of textual data extraction from PDF documents. Its advanced parsing techniques and text processing capabilities enable the module to accurately access and extract content from diverse document formats and structures. It can navigate through intricate document layouts and handle complex linguistic structures, ensuring comprehensive extraction of textual data. By leveraging its versatility and adaptability, this component empowers users to extract valuable insights from a wide range of document sources.

LLM's robust linguistic analysis capabilities enable the question answering component to accurately interpret the semantic meaning behind user queries, allowing it to generate contextually relevant responses. This integration ensures that users obtain the most pertinent information aligned with their intent, enhancing the overall user experience. Furthermore, both modules benefit from continuous advancements and improvements in LLM's capabilities through ongoing research and development efforts. By staying up-to-date with the latest advancements in natural language processing technology, the tool remains at the forefront of efficient information retrieval and analysis from PDF documents. Overall, the integration of Langchain and OpenAI's language model empowers the tool with state-of-the-art linguistic analysis and extraction capabilities, facilitating seamless interactions and enabling users to derive actionable insights effectively from PDF documents.

CHAPTER 4

SEMANTIC DATA RETRIEVAL

The methodology adopted for the development and evaluation of the PDF-query tool is structured around a systematic and iterative approach, prioritizing key principles such as accuracy, efficiency, and user-centric design. In our implementation, we begin with the extraction of text from the PDF document, followed by splitting the text into smaller, manageable chunks using langchain. Subsequently, language embeddings are downloaded from OpenAI to facilitate deeper semantic understanding of the text. The text segments are then indexed using the Facebook AI Similarity Search framework, involving the creation of a FAISS index and the addition of text segments to this index for efficient retrieval. Upon receiving a user query, the system initiates a search through the indexed text segments to retrieve the relevant documents. The question-answering model is loaded next to process the user query and relevant documents, ultimately producing an answer. Finally, the system outputs the answer, completing the process of querying and retrieving information from the indexed text segments.

4.1 Data Processing

The first step in the methodology involves data processing and preparation. PDF documents are preprocessed to extract textual content and normalize formatting inconsistencies. This step also includes text segmentation to break down large documents into smaller, manageable chunks, optimizing the efficiency of subsequent processing steps. Additionally, the textual data undergoes cleaning and normalization to remove noise and standardize terminology, ensuring consistency and accuracy in information retrieval. Specialized techniques, such as tokenization and stemming, may be applied to further enhance the quality of the data.

The preprocessing step includes language-specific processing to handle linguistic nuances and variations across different languages. This involves language detection to identify the primary language of the document and apply language-specific preprocessing techniques tailored to the linguistic characteristics of the text. It also involves data augmentation techniques to enhance the diversity and representativeness of the dataset, especially in scenarios with limited training data or imbalanced distributions. This include techniques such as data synthesis, where new samples are generated by perturbing existing data through transformations or adding noise, thereby enriching the dataset and improving the robustness of the model.

Moreover, data preprocessing plays a crucial role in addressing data bias and fairness concerns by identifying and mitigating biases present in the dataset. This may involve techniques such as bias detection, where statistical metrics are used to quantify disparities in the distribution of data across different demographic groups, and bias mitigation, where corrective measures are applied to mitigate biases and promote fairness in the modeling process. Overall, the data preprocessing and preparation step encompasses a range of techniques and processes aimed at transforming raw PDF data into a clean, structured, and semantically enriched dataset ready for analysis.

4.1.1 Text Extraction and Normalization

This subsection delves into the process of extracting textual content from PDF documents and normalizing formatting inconsistencies. It covers techniques such as PDF parsing and text extraction algorithms to ensure accurate extraction of text while handling various document formats.

The process involves employing sophisticated techniques to accurately extract textual content from PDF files while ensuring consistency and uniformity in the extracted data. PDF parsing algorithms are utilized to interpret the complex structure of PDF documents, identifying text elements and their associated formatting attributes such as font size, style, and alignment. Text extraction algorithms then extract the identified text elements, taking into account factors like embedded fonts and text encoding to ensure accurate retrieval of textual content.

Moreover, the normalization of formatting inconsistencies is essential to standardize the extracted text data for further processing and analysis. PDF documents may exhibit inconsistencies in formatting due to factors such as font variations, layout differences, or encoding issues. Normalization methods are applied to rectify these inconsistencies, aligning the formatting attributes of the extracted text to a standardized format. This process may involve converting text to a consistent font style and size, ensuring uniform text alignment, and resolving encoding discrepancies to maintain the integrity and consistency of the extracted text data.

4.1.2 Text Segmentation for Efficient Processing

This subsection focuses on the segmentation of text within PDF documents to break down large documents into smaller, manageable chunks. It explores techniques for segmenting text based on logical boundaries such as paragraphs, headings, or sections, optimizing the efficiency of subsequent processing steps by enabling targeted analysis of specific content segments.

It’s a crucial step in the data preprocessing phase, particularly when dealing with large and complex PDF documents. Segmentation techniques are employed to divide the textual content within PDF documents into smaller, more manageable chunks based on logical boundaries such as paragraphs, headings, or sections. By breaking down large documents into smaller segments, the efficiency of subsequent processing steps is significantly enhanced, as it allows for targeted analysis of specific content segments without the need to process the entire document at once.

Moreover, segmented text can be indexed and organized more effectively, facilitating faster retrieval of relevant information during search queries or information extraction tasks. Segmentation aids in improving the readability and usability of PDF documents by breaking down lengthy passages into smaller, more digestible chunks, enhancing the overall user experience when interacting with the documents. It plays a vital role in streamlining the analysis and processing of PDF documents within the PDF-query tool. By dividing documents into smaller segments based on logical boundaries, this process optimizes efficiency, enables targeted analysis, and enhances the usability of PDF documents, ultimately contributing to more effective information retrieval and analysis.

The entire process of text segmentation involves :

- Sentence Segmentation: The text is split into sentences or smaller segments, depending on the context and intended use. This segmentation helps isolate individual pieces of information for easier indexing and retrieval.

- Chunking by Length: To maintain consistency and efficiency in processing, the text is divided into chunks of predefined lengths. This approach ensures that each chunk contains a manageable amount of text, reducing the complexity of later steps.

- Metadata Association: Each text chunk is associated with metadata that identifies its source document, page number, and other relevant information. This metadata aids in locating and retrieving specific chunks during query processing.

4.1.3 Cleaning and Normalization of Text Data

This phase addresses the cleaning and normalization of textual data to remove noise and standardize terminology. It discusses techniques such as text cleaning to eliminate irrelevant characters or symbols and normalization methods to standardize terminology and ensure consistency in information retrieval. Additionally, it may cover processes like spell checking and grammar correction to improve the quality of the textual data. This process involves a series of techniques designed to remove noise, standardize terminology, and improve the overall quality of the textual data.

One key aspect of cleaning textual data involves the removal of irrelevant characters or symbols that may be present due to encoding issues, formatting inconsistencies, or artifacts introduced during the text extraction process. Techniques such as regular expressions or string manipulation algorithms are employed to identify and eliminate these extraneous elements, ensuring that the text data is clean and free from any unwanted noise. In addition to removing noise, the normalization of textual data focuses on standardizing terminology and ensuring consistency in information retrieval. This entails identifying variations in spelling, punctuation, or terminology usage within the text and harmonizing them to a standardized format. For example, normalization methods may involve converting all text to lowercase, standardizing abbreviations or acronyms, and resolving synonyms or homographs to ensure uniformity in the representation of terms across the dataset. By standardizing terminology, the PDF-query tool can improve the accuracy and relevance of information retrieval results, as users can expect consistent and predictable outcomes when querying the system.

Spell checking algorithms can identify and correct misspelled words or typographical errors, while grammar correction techniques can rectify grammatical mistakes or syntactic inconsistencies within the text. By addressing these linguistic imperfections, the PDF-query tool can ensure that the extracted textual data is linguistically accurate and grammatically correct, enhancing the overall usability and reliability of the information retrieved from PDF documents.

4.1.4 Language Specific Processing and Structural Analysis

This phase explores language-specific processing techniques to handle linguistic nuances and variations across different languages. It covers language detection methods to identify the primary language of the document and applies language-specific preprocessing techniques tailored to linguistic characteristics. It also discusses structural analysis techniques to capture hierarchical relationships and semantic structure within the text, preserving the significance of structural elements such as headings, paragraphs, and annotations during preprocessing. It’s an essential component of the data preprocessing phase, particularly when dealing with multilingual PDF documents or documents containing content in languages with unique linguistic characteristics.

Language-specific processing techniques are employed to handle linguistic nuances and variations across different languages, ensuring accurate and consistent preprocessing of textual data. It also involves language detection methods, which are utilized to identify the primary language of the document. Language detection algorithms analyze textual patterns, word frequencies, and syntactic structures to determine the language of the document accurately. Once the primary language is identified, language-specific preprocessing techniques tailored to the linguistic characteristics of that language are applied to the textual data. These techniques may include tokenization, stemming, lemmatization, and stop word removal, among others, optimized for the specific linguistic nuances and structures of the identified language. By customizing preprocessing techniques based on the language of the document, the PDF-query tool can ensure that the extracted textual data is accurately processed and prepared for subsequent analysis.

Structural analysis techniques are employed to capture hierarchical relationships and semantic structures within the text, preserving the significance of structural elements such as headings, paragraphs, lists, and annotations during preprocessing. Structural analysis involves parsing the document to identify and extract structural elements, which convey important contextual information and semantic relationships within the text.

The preprocessing phase ensures that the extracted textual data retains its semantic richness and contextual relevance, enabling more precise and insightful analysis results by accounting for linguistic nuances and preserving structural elements. Language-specific preprocessing techniques enhance the tool's adaptability to diverse language environments, allowing it to effectively process and analyze textual data in multiple languages or language variants.

Overall, during text extraction, page-by-page parsing occurs i.e. the text extraction tool parses each page individually, ensuring that all text is captured. This method allows for a thorough examination of each page, including headers, footers, and other elements that may not be part of the main body text. Also, in cases where PDFs contain embedded images, additional tools like Optical Character Recognition (OCR) are used to extract text from images.

The text splitting associated in this phase involves sentence segmentation i.e. the text is split into sentences or smaller segments, depending on the context and intended use. This segmentation helps isolate individual pieces of information for easier indexing and retrieval. To maintain consistency and efficiency in processing, the text is divided into chunks of predefined lengths.

This approach ensures that each chunk contains a manageable amount of text, reducing the complexity of later steps. Also, each text chunk is associated with metadata that identifies its source document, page number, and other relevant information. This metadata aids in locating and retrieving specific chunks during query processing.

4.2 Language Embeddings

Language embeddings play a crucial role in understanding the semantic relationships between words, phrases, and sentences. In this phase, we use OpenAI's language embeddings to encode the extracted text into a form that captures its semantic structure. This step is essential for enabling deeper analysis and comparison of text chunks.

The process of obtaining language embeddings involves:

- Embedding Model Selection: A suitable language embedding model from OpenAI is chosen based on the desired level of semantic understanding and the characteristics of the text. This model encodes text into high-dimensional vectors that represent the underlying meaning of the content.

- Embedding Calculation: Each text chunk is converted into its corresponding vector representation using the selected embedding model. This step transforms the raw text into a format that can be used for semantic similarity analysis and indexing.

- Storage of Embeddings: The calculated embeddings are stored along with their corresponding text chunks and metadata. This storage structure allows for efficient retrieval and indexing in the next step.

4.3 Indexing with FAISS

With the text chunks converted into vector representations, we move to the indexing phase. Facebook AI Similarity Search (FAISS) is employed to create an index that enables efficient retrieval of relevant text chunks based on semantic similarity. FAISS is particularly useful for handling large volumes of high-dimensional data.

Indexing with FAISS involves:

- Index Creation: A FAISS index is created using the stored embeddings. This index allows for rapid search and retrieval based on semantic similarity, reducing the time required to find relevant information.

- Adding Text Segments to the Index: Each text chunk's corresponding embedding is added to the FAISS index. This step ensures that the index contains all the semantic information needed to retrieve relevant text segments when queried.

- Index Optimization: Depending on the expected query load and data volume, additional optimization techniques are applied to the FAISS index to improve search efficiency. This optimization may include adjusting index parameters or using specific search strategies to speed up retrieval.

By the end of this phase, the text chunks have been transformed into semantic embeddings and indexed with FAISS, creating an efficient structure for rapid retrieval of relevant information based on user queries.

4.4 Information Extraction

In this phase, the system retrieves relevant information based on user queries, process the retrieved documents, and extract answers. This phase encompasses the interaction between the user and the system, utilizing the indexed text segments and question-answering models to deliver accurate and relevant answers.

4.4.1 User Query and Search

The process begins when a user submits a query to the system. The query is analyzed to understand its intent and to identify key elements that will guide the retrieval process. The system then initiates a search through the indexed text segments using FAISS to find documents likely to contain relevant information.

Steps in the search process include:

- Query Analysis: The user query is parsed and analyzed to determine the key terms and phrases that define the search criteria. This analysis may involve natural language understanding techniques to ensure the system understands the user's intent.

- FAISS Search: The system uses the parsed query to search through the FAISS index, retrieving text segments with semantic similarity to the query. This retrieval step aims to find documents that are most likely to contain answers or relevant information.

- Document Selection: The retrieved text segments are evaluated to select those that best match the query. This selection process ensures that the system focuses on the most relevant documents, improving the efficiency of the question-answering step.

4.4.2 Answer Retrieval

With the relevant text segments identified, the system proceeds to the question-answering phase. A question-answering model is loaded to process both the user query and the content of the retrieved documents. This model uses advanced natural language processing techniques to extract answers from the documents.

Steps in the question-answering process include:

- Model Loading: The question-answering model is loaded and prepared for use. This model employs pre-trained deep learning architectures designed for question-answering tasks.

- Processing of Relevant Documents: The model processes each relevant document in the context of the user query. It analyzes the text segments to identify sections that contain potential answers to the query.

- Answer Extraction: Based on the model's analysis, the system extracts information that likely answers the user's query. This extraction process may involve selecting specific text passages or generating a summary that addresses the query.

4.4.3 Answer Validation and Output

After extracting answers from the relevant documents, the system performs a validation step to ensure the relevance and correctness of the extracted information. This validation process aims to confirm that the answer accurately addresses the user's query and is reliable.

Steps in the validation and output process include:

- Relevance Check: The extracted information is checked for relevance to the user query. This step ensures that the final answer is closely aligned with the query's intent.

- Correctness Check: The system verifies the correctness of the extracted information, ensuring that it is accurate and consistent with the source documents.

- Final Answer Compilation: After scanning all relevant documents and validating the extracted information, the system compiles the final answer to the user query. This answer may be presented in a suitable format, such as a concise summary or a direct response.

By the end of this phase, we have successfully retrieved relevant information, processed it using a question-answering model, validated the extracted answers, and delivered a reliable response to the user query.

The algorithm covered below is an acute representation of the methodology of our proposed work.

```
Require: User query, $\mathcal{D}$ is the set of documents
Ensure: $\mathcal{F}$ is the final answer to the user query
Function Information Extraction ($\mathcal{D}$)
$\mathcal{F} := \emptyset$
For each document $\mathcal{D}_i$ in $\mathcal{D}$
    Extract text from $\mathcal{D}_i$
    Split text into smaller segments using langchain
    Download language embeddings from OpenAI
    Index text segments using FAISS
Receive user query
```

```
Search indexed text segments for relevant documents
Load question-answering model
For each relevant document $\mathcal{D}_i$
Process user query and document using question-answering model
Check the relevance and correctness of the extracted information from the document
Obtain $\mathcal{F}$(i.e. the final answer) after scanning all the releveant documents
Return $\mathcal{F}$
```

Firstly, the algorithm depicts that each document is processed in the document set. The text from each document is extracted and broken into smaller fragments for better handling. Then, indices for these text segments are created using a technique called FAISS (Facebook AI Similarity Search). This allows the quick and faster search for relevant segments when given a user query. When a user submits a query, the search is initiated through the indexed text segments to find documents that are likely to contain relevant information related to the query. For each relevant document found, a question-answering model is used to process both the user query and the document's content. After extracting information from each relevant document, the relevance and correctness of the extracted information is checked subsequently. It ensures that the information accurately addresses the user's query and is reliable.

On inspecting the code, firstly from PyPDF2, PdfReader is imported, enabling the code to read and process text from PDF documents.From the langchain package, various components are imported for handling text data and embeddings i.e. OpenAIEmbeddings from the embeddings module indicates the utilization of embeddings provided by OpenAI, while CharacterTextSplitter from the text_splitter module may be used to break down text into smaller units, potentially characters.

Classes related to vector storage and retrieval from the langchain package's vectorstores module are also imported. Classes, such as ElasticVectorSearch, Pinecone, Weaviate, and FAISS, provide interfaces for interacting with different vector stores or databases, which are crucial for storing and querying vector representations of text data efficiently.
Then, a function called sentence_bleu is imported from the nltk package's bleu_score module. This function computes the BLEU score, a metric commonly used to evaluate

the quality of machine-translated text against one or more reference translations. It's a useful tool for assessing the performance of translation systems in NLP applications. So, we set up an environment equipped with the necessary tools and libraries for a wide range of NLP tasks, including text extraction from PDFs, working with embeddings, managing text data, storing and retrieving vector representations efficiently, and evaluating machine translation quality.

Then, the os module is imported , which is a part of Python's standard library, to interact with the operating system. Specifically, it utilizes the os.environment dictionary-like object to set the environment variable OPENAI_API_KEY to a provided API key string. This step is crucial for authentication when making requests to OpenAI's API, ensuring that subsequent interactions with OpenAI's services are properly authorized.

Following this, our Google Drive account is connected to a Google Colab notebook environment, allowing access to files stored in Google Drive. After mounting the Google Drive, it defines a variable root_dir to represent the root directory of the mounted drive. Then, it uses the PdfReader class from the PyPDF2 package to read a specific PDF file located within the mounted Google Drive. This enables seamless access and processing of PDF files stored in Google Drive directly from the Colab notebook environment, streamlining tasks such as data analysis or document processing.

Subsequently, a CharacterTextSplitter object is initialized to break down a large text into smaller chunks, mitigating token size limitations during information retrieval processes. The CharacterTextSplitter is configured with parameters including a separator character (in this case, a newline \n), a chunk size of 1000 characters, a chunk overlap of 200 characters, and a length function.

These settings ensure that the text is divided into manageable portions while allowing for some overlap between chunks. Finally, the split_text() method is called on the text_splitter object to split the raw text into smaller chunks based on the specified parameters, resulting in a list of text chunks ready for further processing or analysis.

Then, embeddings from OpenAI are downloaded and a document search object is initialized using FAISS, a library for similarity search and clustering of dense vectors.

The OpenAIEmbeddings() call retrieves embeddings from OpenAI, while FAISS.from_texts(texts, embeddings) constructs a FAISS index from the provided texts and embeddings. This index allows for efficient searching and retrieval of similar documents based on their semantic similarities, enabling various text retrieval tasks such as document similarity analysis or information retrieval.

We utilize the langchain library to perform a question-answering task. The system first loads a question-answering chain using the load_qa_chain() function from the langchain.chains.question_answering module, specifying OpenAI's language model (LLM) as the underlying model. The chain type is set to "stuff", which indicates a general-purpose question-answering model. Subsequently, a query is formulated with the variable query, asking about the distribution of LeadSquare's workforce across different parts of the world. Then, the docsearch object, likely containing pre-indexed documents with embeddings, performs a similarity search to retrieve relevant documents based on the query.

Finally, the question-answering chain is executed with chain.run(), taking the retrieved documents and the query as input. This process leverages the language model to analyze the documents and generate an answer to the query. The quality of machine-generated text is evaluated by calculating BLEU scores, a metric commonly used in natural language processing tasks. It compares generated text against reference text and computes BLEU scores for each pair.

Then, the average BLEU score across all pairs, are calculated providing a single metric to quantify the overall performance of the text generation model. This approach offers a systematic and quantitative assessment of the model's effectiveness in producing text that closely matches the desired reference.

Ultimately, the performance of a language model is evaluated by comparing its generated responses to ground truth responses using precision, recall, and F1-score metrics. It first imports necessary functions from NLTK and scikit-learn libraries for tokenization and metric calculation. Then, it defines ground truth responses and responses generated by the language model. These responses are tokenized into lists of words.

CHAPTER 5

RESULTS AND DISCUSSIONS

5.1 Effect of Chunk Size

In our research, the entire text of the document is split into chunks. Here, chunk size plays a crucial role in dividing the document into manageable segments. The choice of chunk size is a balance between granularity and efficiency. With a chunk size set to 1000 characters, we ensure that each segment is of a manageable length for processing. Larger chunk sizes may lead to better computational efficiency but also results in the loss of fine-grained details. Conversely, smaller chunk sizes provide more granularity but increases computational overhead. By setting the chunk_size parameter to 1000, we aim to strike a balance between granularity and efficiency, ensuring effective processing while retaining important contextual information within each segment. Through our proposed approach, we try to ensure that the system delivers accurate and reliable results, particularly in tasks such as information extraction and question answering, where precision is crucial.

We have also used a crucial parameter called chunk_overlap in our approach. The chunk_overlap parameter determines the overlap between consecutive chunks of text. Overlapping chunks play a crucial role in maintaining context and coherence during text analysis. In our research, we set the chunk_overlap parameter to 200, meaning that adjacent chunks overlap by 200 characters. This ensures that important contextual information is not lost at chunk boundaries. Overlapping chunks enable a smoother transition between segments, allowing for a more seamless analysis of the text. By reducing the risk of information loss, overlapping chunks contribute to the overall effectiveness of the text analysis process. Through the integration of Langchain and OpenAI's Language Model, our proposed approach demonstrates effectiveness in extracting textual data, converting it into numerical vectors for semantic analysis, and generating accurate and context-aware responses.

Chunk Size	Precision	Recall	F1-Score
1000	0.75	0.60	0.67
500	0.70	0.65	0.68

Table 4.1 Experimental Results at varied chunk sizes

The inference which can be drawn from Table 5.1 is, on performing experimental analysis, we obtain interesting insights on the impact of chunk size on the performance of the system. With a chunk size of 1000, the system achieved higher precision but lower recall compared to a chunk size of 500. This suggests that larger chunk sizes leads to more accurate answers but at the expense of potentially missing out on relevant information. On the other hand, smaller chunk sizes offer a better balance between precision and recall, resulting in a slightly higher F1-score. These findings highlight the importance of carefully tuning the chunk size parameter to optimize system performance based on specific use cases and requirements. The performance metrics at different chunk sizes is depicted in Figure 5.1.

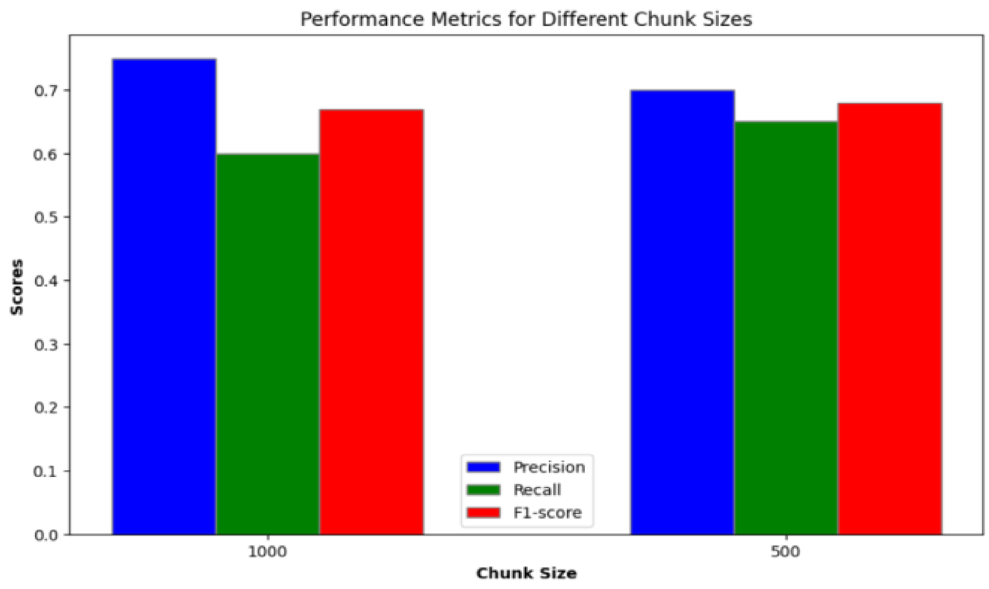


Fig 5.1 Performance metrics at varied chunk sizes

In our approach, we emphasize precision because it reflects the accuracy of the system in providing correct answers among all the answers it provides. Precision is particularly crucial when dealing with tasks such as information extraction or question answering, where the goal is to ensure that the provided answers are as accurate as possible. Choosing a chunk size of 1000 over 500 is based on the balance between precision and other metrics such as recall and F1-score. While a chunk size of 500 showed slightly better recall and F1-score, the higher precision achieved with a chunk size of 1000 indicates that the system is more accurate in providing correct answers. In scenarios where accuracy and precision are paramount, such as information retrieval from PDF documents or generating summaries, prioritizing higher precision can be beneficial. Therefore, in our proposed work, we choose a chunk size of 1000 to maximize precision and ensure that the system delivers accurate and reliable results.

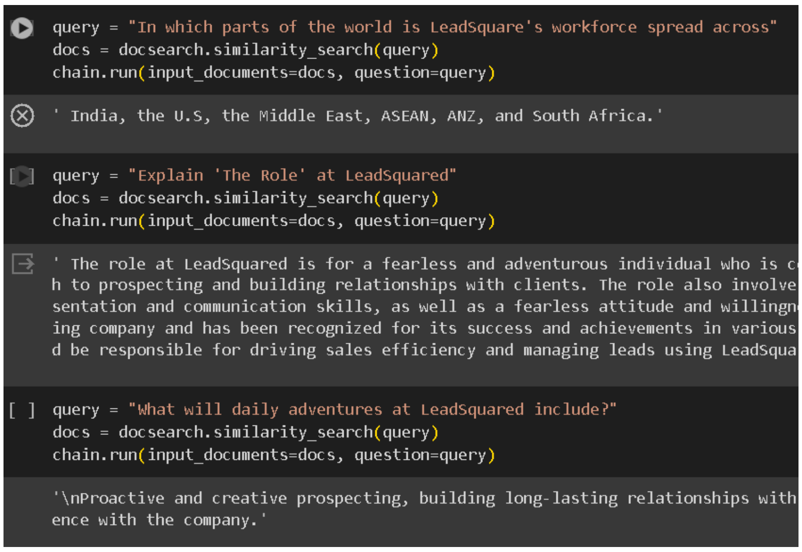
5.2 BLEU Metric

Moreover, in our work, we have also implemented another evaluation metric which is far more suitable comapred to precision, recall etc. for text genreation tasks where the entire response to the user query is generated. This performance metric is known as the BLEU (Bilingual Evaluation Understudy) score. This score serves as a valuable metric for evaluating the quality of machine-generated text. In our research, we utilize the BLEU score to assess the quality of answers generated by the OpenAI Language Model.

By comparing machine-generated answers with human translation references (generated responses and ground-truth responses), the BLEU score provides a quantitative measure of grammatical correctness and factual accuracy. Higher BLEU scores indicate a closer alignment with human references, reflecting the overall effectiveness of the generated responses. Through the use of the BLEU score, we gain valuable insights into the performance of the OpenAI Language Model in generating accurate and contextually relevant answers. The BLEU score gives an output score between 0 and 1. A BLEU score of 1 depicts that the sentence perfectly matches one of the reference sentences.

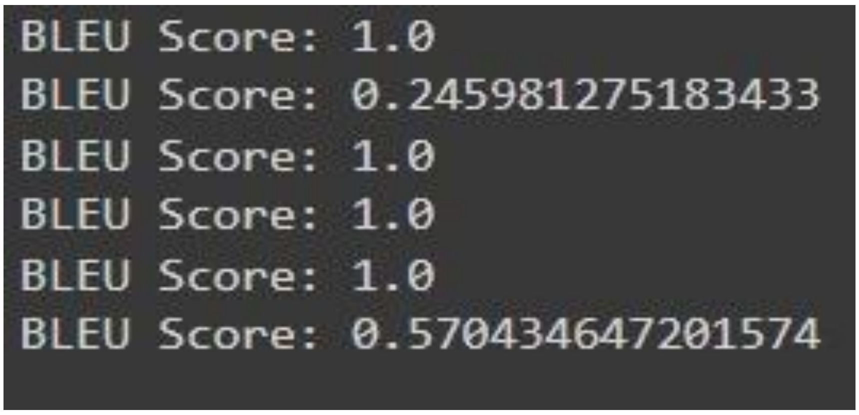
The experimental results demonstrate the system's ability to accurately comprehend and analyze documents, delivering relevant information based on user-defined prompts. Through qualitative evaluation, we assessed the system's performance in providing precise summaries tailored to users' requirements.

When responding to user queries, the tool utilizes a question-answering model loaded from OpenAI to generate answers based on the retrieved documents.



```
query = "In which parts of the world is LeadSquare's workforce spread across"
docs = docsearch.similarity_search(query)
chain.run(input_documents=docs, question=query)

' India, the U.S, the Middle East, ASEAN, ANZ, and South Africa.'

query = "Explain 'The Role' at LeadSquared"
docs = docsearch.similarity_search(query)
chain.run(input_documents=docs, question=query)

' The role at LeadSquared is for a fearless and adventurous individual who is c
h to prospecting and building relationships with clients. The role also involve
sentation and communication skills, as well as a fearless attitude and willingn
ing company and has been recognized for its success and achievements in various
d be responsible for driving sales efficiency and managing leads using LeadSqua

[ ] query = "What will daily adventures at LeadSquared include?"
docs = docsearch.similarity_search(query)
chain.run(input_documents=docs, question=query)

'\nProactive and creative prospecting, building long-lasting relationships with
ence with the company.'
```

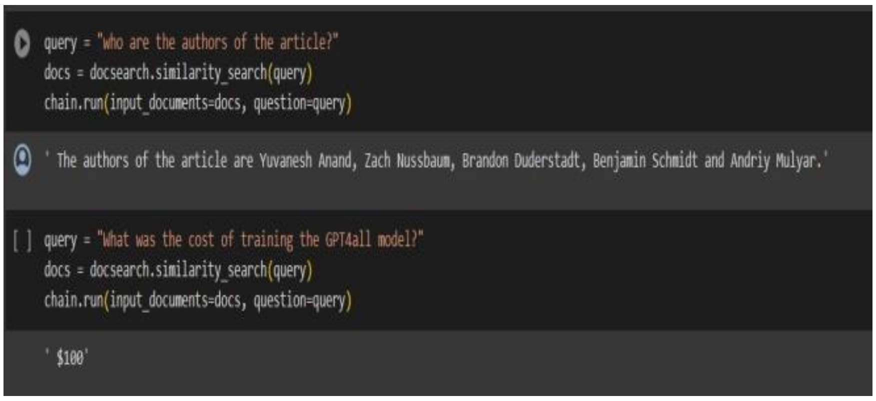
Fig 5.2 User Queries and Generated Responses



```
BLEU Score: 1.0
BLEU Score: 0.245981275183433
BLEU Score: 1.0
BLEU Score: 1.0
BLEU Score: 1.0
BLEU Score: 0.570434647201574
```

Fig 5.3 BLEU scores corresponding to various responses

The prompts and the genrated responses, by the LLM are shown in Figure 5.2 and Figure 5.3, and the BLEU scores corresponding to the genrated responses are also shown in the same. The BLEU score that we have used in our approach is better than the evaluation metrics used in other realted works in certain aspects. This performance metric is more relevant since our work significantly reformulates retrieved information to create the answer, with grammatical correctness alongside factual accuracy. For open ended questions where the answer can be phrased in multiple ways, BLEU provides insight into how well the answer captures the overall meaning of the relevant retrieved information.



```
query = "who are the authors of the article?"
docs = docsearch.similarity_search(query)
chain.run(input_documents=docs, question=query)
```

```
' The authors of the article are Yuvanesh Anand, Zach Nussbaum, Brandon Duderstadt, Benjamin Schmidt and Andriy Mulyar.'
```

```
[ ] query = "What was the cost of training the GPT4all model?"
docs = docsearch.similarity_search(query)
chain.run(input_documents=docs, question=query)
```

```
' $100'
```

Fig 5.4 Queries and corresponding responses to data in documents

In Figure 5.4, the queries and the responses for the specific texts are shown. Also, the responses are limited to information contained only in the document. For example: When asked " What is Google Bard? ", the system responds with "I don't know."

This indicates that the document does not contain information about Google Bard, and therefore, it cannot provide a meaningful response, which can be seen below. Our proposed approach leverages Langchain and OpenAI's Language Model to effectively extract and analyze information from PDF documents. Experimental results demonstrate the tool's effectiveness in extracting textual data, converting it into numerical vectors for semantic analysis, and indexing processed text segments for efficient retrieval. Through qualitative evaluation, we assessed the tool's performance in providing precise summaries tailored to users' requirements, leveraging state-of-the-art language models for accurate and context-aware responses.

The integration of Langchain and OpenAI's Language Model enhances the tool's capabilities, enabling fast and accurate retrieval of relevant information and generating accurate answers based on retrieved documents. Compared to traditional models, this approach benefits from the advanced capabilities of deep learning models, contributing to the development of a robust and efficient information retrieval system.

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENTS

6.1 Conclusion

According to the experimental results, our approach excels in handling PDF documents, which are a common format for storing and sharing text-based information. Furthermore, our use of the BLEU score as a metric for evaluating the quality of machine-generated text provides valuable insights into the performance of the OpenAI Language Model. The BLEU score allows us to assess the grammatical correctness and factual accuracy of the generated answers, ensuring that the responses align closely with human references. The primary limitation of our work is its ability to processing only PDF documents. This restriction may hinder its applicability in scenarios where information is stored in other formats, such as Word documents, HTML files, or structured databases. Users may need to convert their data to PDF format before utilizing the system, leading to inconvenience and potential data loss. These limitations will be looked into and resolved later.

6.2 Future Enhancements

In the near future, the following enhancements will be implemented in our work:

- Development of a deployment strategy will be conducted that ensures the efficient deployment of the PDF-query tool in the intended environment. Containerization technologies like Docker will be utilized to package the application and its dependencies for seamless deployment. The system's performance will be optimized by implementing caching mechanisms, parallel processing techniques, and query optimization strategies to enhance response times and resource utilization.

- Robust monitoring and logging mechanisms will be implemented to track system performance metrics in real-time. Utilization of monitoring tools and dashboards will also be carried out to visualize system performance and identify potential bottlenecks or performance issues. The process to continuously analyze performance metrics and conduct performance tuning exercises to optimize system parameters and configurations for maximum efficiency, will also be performed.

- Lastly, implementation of comprehensive security measures will be executed to protect sensitive data and ensure compliance with data privacy regulations, utilize encryption techniques to secure data both in transit and at rest. Access control mechanisms will be implemented to restrict access to the PDF-query tool and its underlying data only to authorized users. Also, regular conduction of security audits and vulnerability assessments will be done to identify and address potential security vulnerabilities.

REFERENCES

[1] Khin, N. T. W., & Yee, N. N. (2018, October). Query classification based information retrieval system. In *2018 International conference on intelligent informatics and biomedical sciences (ICIIBMS)* (Vol. 3, pp. 151-156). IEEE.

[2] Nahm, U. Y., & Mooney, R. J. (2002, March). Text mining with information extraction. In *Proceedings of the AAAI 2002 Spring Symposium on Mining Answers from Texts and Knowledge Bases* (pp. 60-67). Stanford CA.

[3] Tari, L., Tu, P. H., Hakenberg, J., Chen, Y., Son, T. C., Gonzalez, G., & Baral, C. (2010). Incremental information extraction using relational databases. *IEEE Transactions on Knowledge and Data Engineering*, *24*(1), 86-99.

[4] Bastian, M. R., & Purwarianti, A. (2016, October). Information extraction in statistics indicator tables using rule generalizations and ontology. In *2016 International Conference on Information Technology Systems and Innovation (ICITSI)* (pp. 1-6). IEEE.

[5] Liu, S., & Ren, F. (2011, September). Paragraph act based pragmatic information extraction in question answering. In *2011 IEEE International Conference on Cloud Computing and Intelligence Systems* (pp. 153-157). IEEE.

[6] Rasheed, I., & Banka, H. (2018, March). Query expansion in information retrieval for Urdu language. In *2018 Fourth International Conference on Information Retrieval and Knowledge Management (CAMP)* (pp. 1-6). IEEE.

[7] Saste, R. P., & Patil, S. S. (2014, August). Extraction of incremental information using query evaluator. In *2014 First International Conference on Networks & Soft Computing (ICNSC2014)* (pp. 324-328). IEEE.

[8] Feilmayr, C. (2011, August). Text mining-supported information extraction: an extended methodology for developing information extraction systems. In *2011 22nd International Workshop on Database and Expert Systems Applications* (pp. 217-221). IEEE.

[9] Kamp, S., Fayazi, M., Benameur-El, Z., Yu, S., & Dreslinski, R. (2023). Open Information Extraction: A Review of Baseline Techniques, Approaches, and Applications. *arXiv preprint arXiv:2310.11644*.

[10] Ferrucci, D., & Lally, A. (2004). UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, *10*(3-4), 327-348. [11] Cunningham, H. (2002). GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proc. 40th annual meeting of the association for computational linguistics (ACL 2002)* (pp. 168-175).

[12] Chen, F., Doan, A., Yang, J., & Ramakrishnan, R. (2008, April). Efficient information extraction over evolving text data. In *2008 IEEE 24th International Conference on Data Engineering* (pp. 943-952). IEEE.

[13] Sarawagi, S. (2008). Information extraction. *Foundations and Trends® in Databases*, *1*(3), 261-377.

[14] Leaman, R., & Gonzalez, G. (2008). BANNER: an executable survey of advances in biomedical named entity recognition. In *Biocomputing 2008* (pp. 652-663).

[15] Cafarella, M. J., & Etzioni, O. (2005, May). A search engine for natural language applications. In *Proceedings of the 14th international conference on World Wide Web* (pp. 442-452).

[16] Cheng, T., & Chang, K. C. C. (2007). *Entity Search Engine: Towards Agile Best-Effort Information Integration over the Web* (Doctoral dissertation, University of Illinois at Urbana-Champaign).

[17] Agichtein, E., & Gravano, L. (2000, June). Snowball: Extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital libraries* (pp. 85-94).

[18] Yue, W., Chen, Z., Lu, X., Lin, F., & Liu, J. (2005, November). Using query expansion and classification for information retrieval. In *2005 First International Conference on Semantics, Knowledge and Grid* (pp. 31-31). IEEE.

[19] Fathalla, S. M., Hassan, Y. F., & El-Sayed, M. (2012, October). A hybrid method for user query reformation and classification. In *2012 22nd International Conference on Computer Theory and Applications (ICCTA)* (pp. 132-138). IEEE.

[20] Xia, C., & Wang, X. (2015, September). Graph-based web query classification. In *2015 12th web information system and application conference (WISA)* (pp. 241-244). IEEE.

[21] VRL, N. (2009, December). An unsupervised approach to domain-specific term extraction. In *Australasian Language Technology Association Workshop 2009* (p. 94).

[22] Naing, M. M. T. (2013). Ontology-Based Web Query Classification for Research Paper Searching. *International Journal of Innovations in Engineering and Technology (IJIET)*, *2*(1).

[23] Katariya, A., & Pandya, M. (2015). Ontology-Based Web Query Classification. *International Journal of Engineering Research and General Science*, *3*(3), 806-813.

[24] Jansen, B. J., Booth, D. L., & Spink, A. (2008). Determining the informational, navigational, and transactional intent of Web queries. *Information Processing & Management*, *44*(3), 1251-1266.

[25] Cross, V. (1994). Fuzzy information retrieval. *Journal of Intelligent Information Systems*, *3*, 29-56. [26] Mughal, M. J. H. (2018). Data mining: Web data mining techniques, tools and algorithms: An overview. *International Journal of Advanced Computer Science and Applications*, *9*(6).

[27] Singhal, A. (2001). Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, *24*(4), 35-43.

[28] Alnofaie, S., Dahab, M., & Kamal, M. (2016). A novel information retrieval approach using query expansion and spectral-based. *International Journal of Advanced Computer Science and Applications*, *7*(9).

[29] Dahab, M. Y., Kamel, M., & Alnofaie, S. (2017). Further investigations for documents information retrieval based on DWT. In *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2016 2* (pp. 3-11). Springer International Publishing.

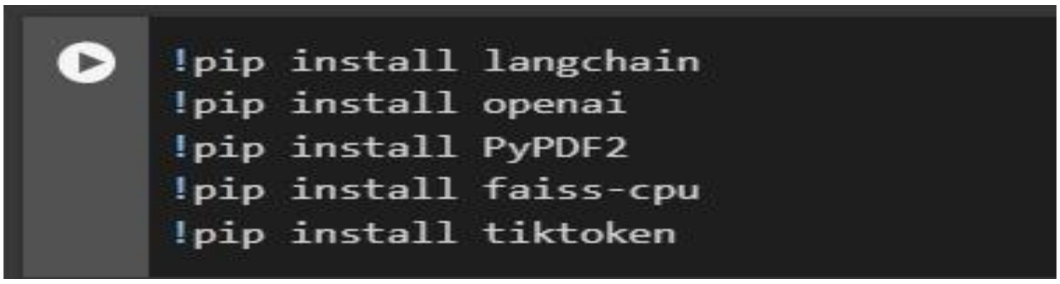
[30] Pal, D., Mitra, M., & Datta, K. (2013). Query expansion using term distribution and term association. *arXiv preprint arXiv:1303.0667*.

[31] Pal, D., Mitra, M., & Datta, K. (2014). Improving query expansion using WordNet. *Journal of the Association for Information Science and Technology*, *65*(12), 2469-2478.

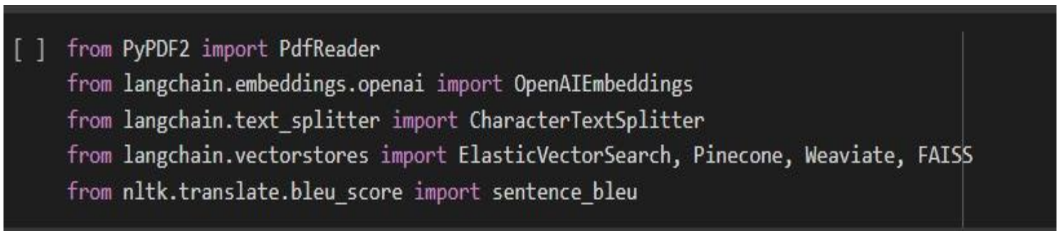
[32] Spink, A., Wolfram, D., Jansen, M. B., & Saracevic, T. (2001). Searching the web: The public and their queries. *Journal of the American society for information science and technology*, *52*(3), 226-234.

APPENDIX A

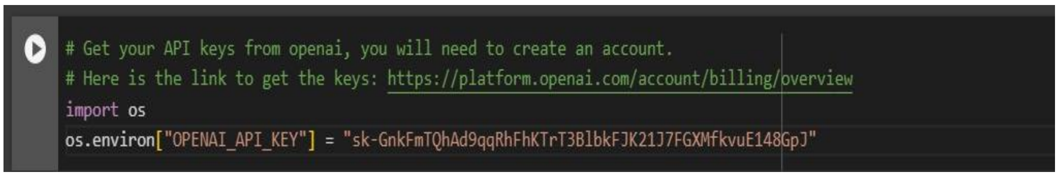
CODING AND TESTING



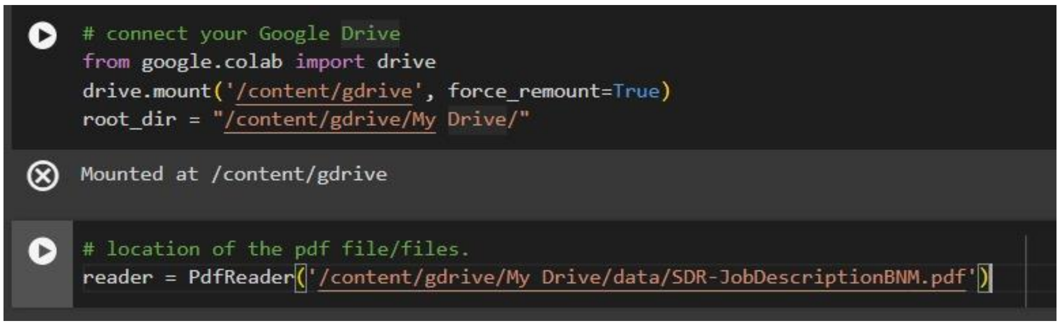
```
!pip install langchain
!pip install openai
!pip install PyPDF2
!pip install faiss-cpu
!pip install tiktoken
```



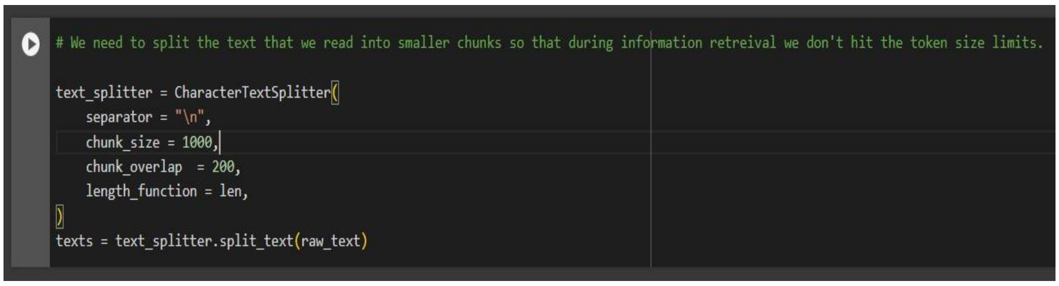
```
[ ] from PyPDF2 import PdfReader
    from langchain.embeddings.openai import OpenAIEmbeddings
    from langchain.text_splitter import CharacterTextSplitter
    from langchain.vectorstores import ElasticVectorSearch, Pinecone, Weaviate, FAISS
    from nltk.translate.bleu_score import sentence_bleu
```



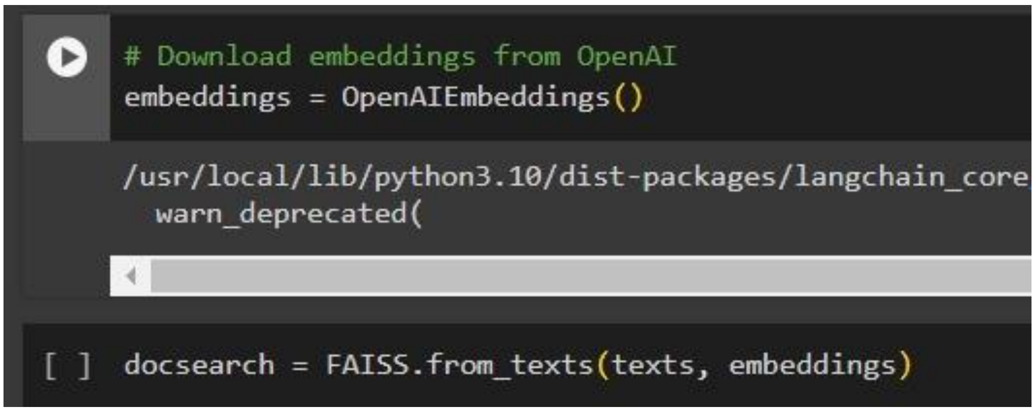
```
# Get your API keys from openai, you will need to create an account.
# Here is the link to get the keys: https://platform.openai.com/account/billing/overview
import os
os.environ["OPENAI_API_KEY"] = "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"
```



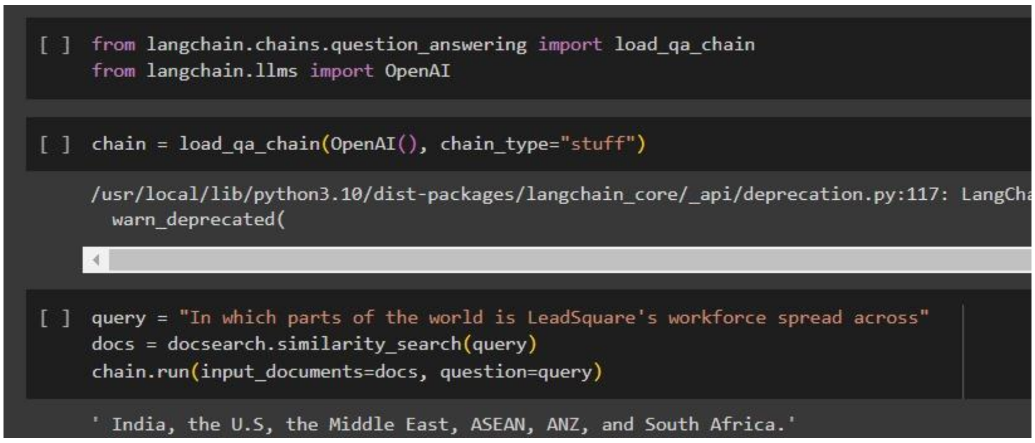
```
# connect your Google Drive
from google.colab import drive
drive.mount('/content/gdrive', force_remount=True)
root_dir = "/content/gdrive/My Drive/"

Mounted at /content/gdrive

# location of the pdf file/files.
reader = PdfReader('/content/gdrive/My Drive/data/SDR-JobDescriptionBNM.pdf')
```



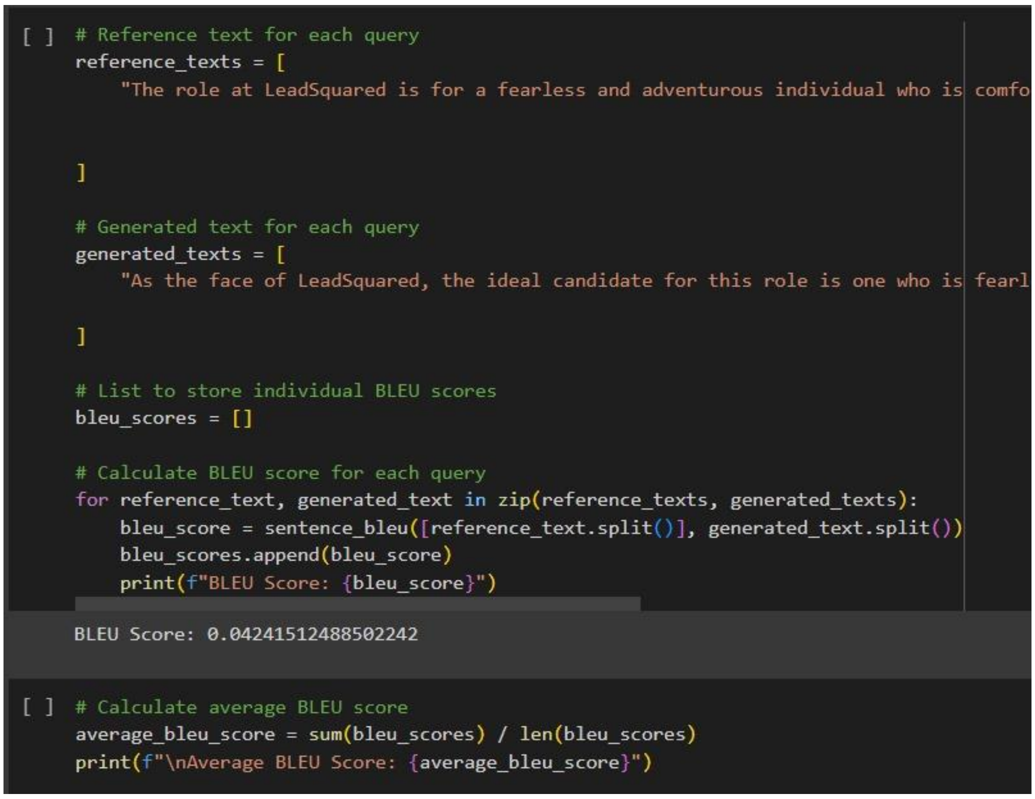
```
# We need to split the text that we read into smaller chunks so that during information retreival we don't hit the token size limits.

text_splitter = CharacterTextSplitter(
    separator = "\n",
    chunk_size = 1000,
    chunk_overlap  = 200,
    length_function = len,
)
texts = text_splitter.split_text(raw_text)
```



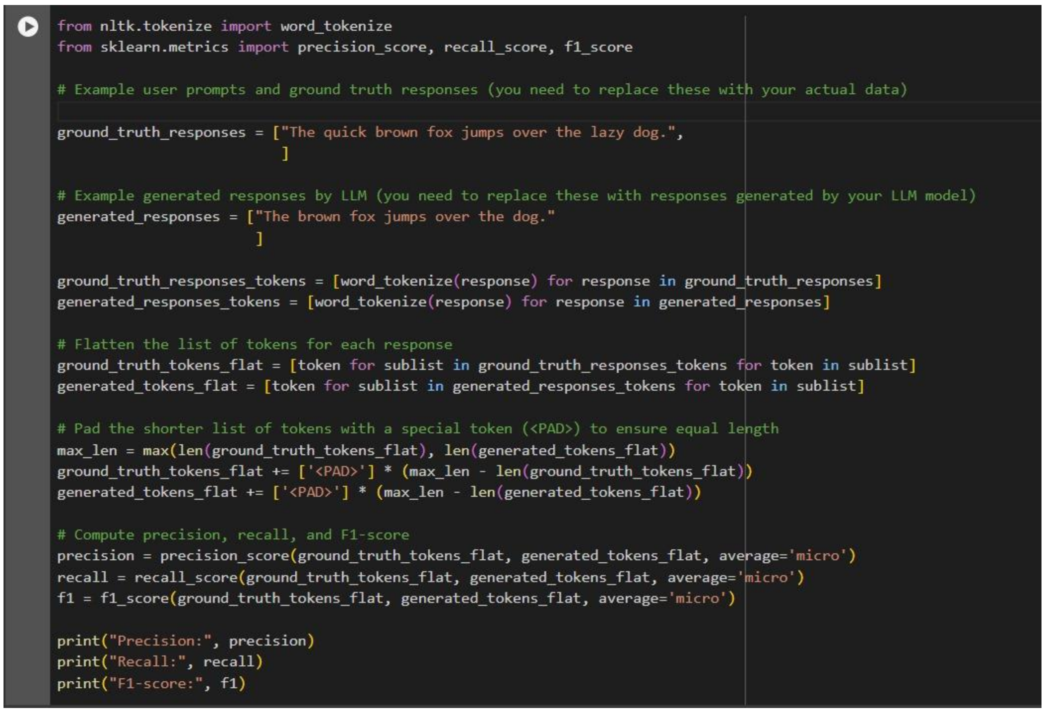
```
# Download embeddings from OpenAI
embeddings = OpenAIEmbeddings()

/usr/local/lib/python3.10/dist-packages/langchain_core
  warn_deprecated(

[ ] docsearch = FAISS.from_texts(texts, embeddings)
```



```
[ ] from langchain.chains.question_answering import load_qa_chain
    from langchain.llms import OpenAI

[ ] chain = load_qa_chain(OpenAI(), chain_type="stuff")

    /usr/local/lib/python3.10/dist-packages/langchain_core/_api/deprecation.py:117: LangCha
      warn_deprecated(

[ ] query = "In which parts of the world is LeadSquare's workforce spread across"
    docs = docsearch.similarity_search(query)
    chain.run(input_documents=docs, question=query)

    ' India, the U.S, the Middle East, ASEAN, ANZ, and South Africa.'
```



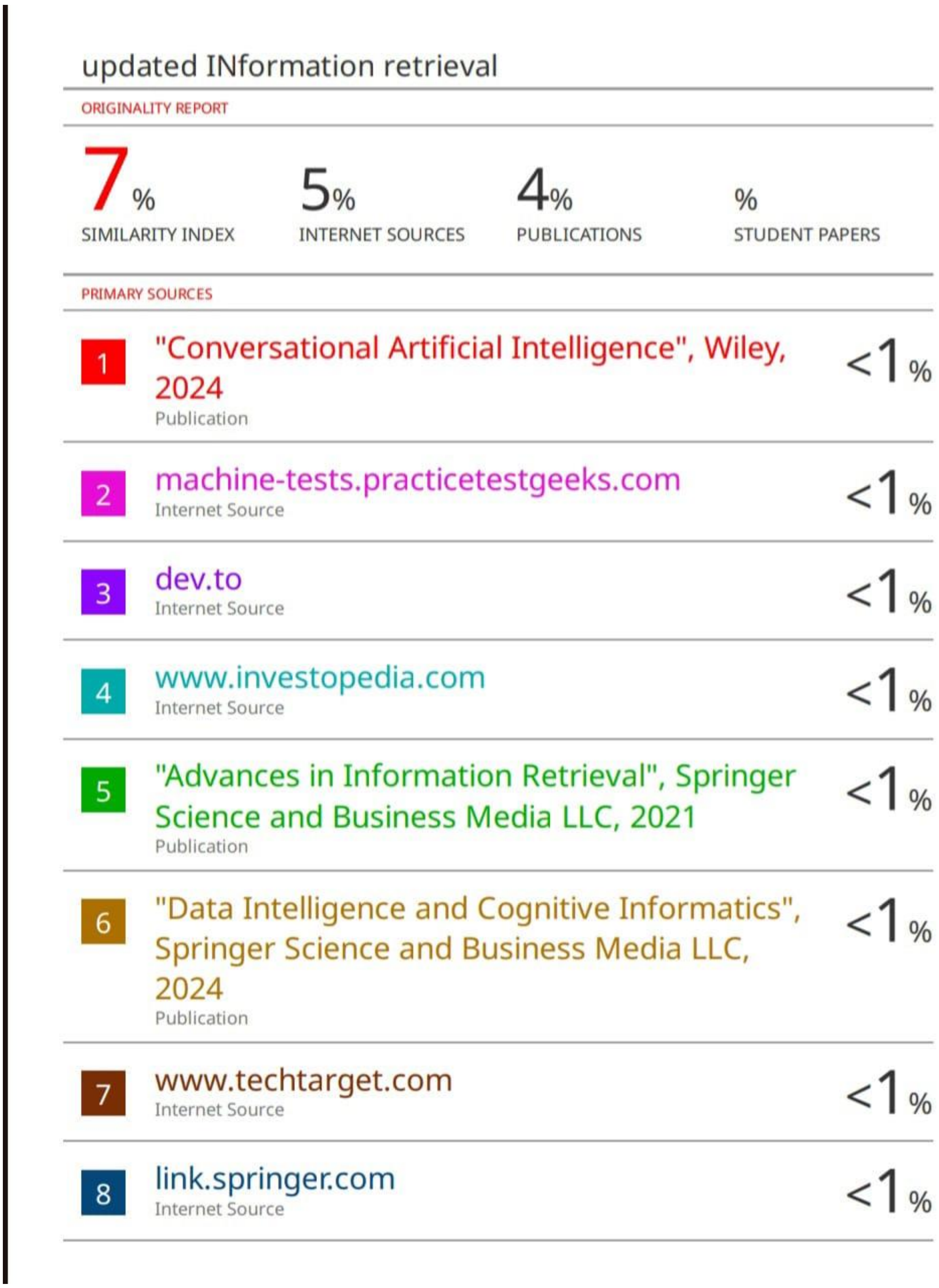
```
[ ] # Reference text for each query
    reference_texts = [
        "The role at LeadSquared is for a fearless and adventurous individual who is comfo


    ]

    # Generated text for each query
    generated_texts = [
        "As the face of LeadSquared, the ideal candidate for this role is one who is fearl

    ]

    # List to store individual BLEU scores
    bleu_scores = []

    # Calculate BLEU score for each query
    for reference_text, generated_text in zip(reference_texts, generated_texts):
        bleu_score = sentence_bleu([reference_text.split()], generated_text.split())
        bleu_scores.append(bleu_score)
        print(f"BLEU Score: {bleu_score}")

    BLEU Score: 0.04241512488502242

[ ] # Calculate average BLEU score
    average_bleu_score = sum(bleu_scores) / len(bleu_scores)
    print(f"\nAverage BLEU Score: {average_bleu_score}")
```



```python
from nltk.tokenize import word_tokenize
from sklearn.metrics import precision_score, recall_score, f1_score

# Example user prompts and ground truth responses (you need to replace these with your actual data)

ground_truth_responses = ["The quick brown fox jumps over the lazy dog.",
                          ]

# Example generated responses by LLM (you need to replace these with responses generated by your LLM model)
generated_responses = ["The brown fox jumps over the dog."
                       ]

ground_truth_responses_tokens = [word_tokenize(response) for response in ground_truth_responses]
generated_responses_tokens = [word_tokenize(response) for response in generated_responses]

# Flatten the list of tokens for each response
ground_truth_tokens_flat = [token for sublist in ground_truth_responses_tokens for token in sublist]
generated_tokens_flat = [token for sublist in generated_responses_tokens for token in sublist]

# Pad the shorter list of tokens with a special token (<PAD>) to ensure equal length
max_len = max(len(ground_truth_tokens_flat), len(generated_tokens_flat))
ground_truth_tokens_flat += ['<PAD>'] * (max_len - len(ground_truth_tokens_flat))
generated_tokens_flat += ['<PAD>'] * (max_len - len(generated_tokens_flat))

# Compute precision, recall, and F1-score
precision = precision_score(ground_truth_tokens_flat, generated_tokens_flat, average='micro')
recall = recall_score(ground_truth_tokens_flat, generated_tokens_flat, average='micro')
f1 = f1_score(ground_truth_tokens_flat, generated_tokens_flat, average='micro')

print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)
```

APPENDIX B

PLAGIARISM REPORT



updated INformation retrieval

ORIGINALITY REPORT

7%	5%	4%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	"Conversational Artificial Intelligence", Wiley, 2024 Publication	<1%
2	machine-tests.practicetestgeeks.com Internet Source	<1%
3	dev.to Internet Source	<1%
4	www.investopedia.com Internet Source	<1%
5	"Advances in Information Retrieval", Springer Science and Business Media LLC, 2021 Publication	<1%
6	"Data Intelligence and Cognitive Informatics", Springer Science and Business Media LLC, 2024 Publication	<1%
7	www.techtarget.com Internet Source	<1%
8	link.springer.com Internet Source	<1%