People's Democratic Republic of Algeria
الجمهورية الجزائرية الديمقراطية الشعبية
Ministry of Higher Education and Scientific Research
وزارة التعليم العالي و البحث العلمي

YAHIA FARÈS UNIVERSITY OF MEDEA
جامعة يحيى فارس-المدية
FACULTY OF SCIENCES
DEPARTMENT OF MATHEMATICS AND
COMPUTER SCIENCE



Master thesis

For the attainment of the Master's degree in Computer Science

Option : System Engineering and Web Technologies

Design and development of a medical assistant based on large language models

Produced by :
BOUSSOUAR Missoum
ZAOUIDI Mahieddine

Supervised by :
Dr. RAHIM Messaoud
Co-Supervised by :
Dr. LALOUANI Wassila

Promotion : 2023/2024

Dedication

This thesis is dedicated to:
My beloved grandfather, , for your unwavering support, invaluable wisdom, and the countless hours you spent mentoring me. Your passion for learning has been an inspiration throughout my academic journey.
My dear grandmother, , for your unconditional love, endless encouragement, and the warmth you have brought to my life. Your resilience and kindness have been a constant source of strength.
My wonderful father, for your guidance, tough love, and belief in my abilities. Your dedication to our family has shaped the person I am today.
My caring mother, for your unconditional support, unwavering patience, and the sacrifices you've made to provide me with every opportunity. Your compassion and nurturing spirit have been instrumental in my success.
My supportive brother,, for your camaraderie, healthy competition, and the many memories we've shared. Your presence in my life has enriched it in ways I cannot fully express.
And to my dear friends, who have become an extended family. Thank you for your friendship, laughter, and for being there for me through the highs and lows. Your presence in my life has made this journey all the more meaningful.
I am truly grateful to have you all in my life, and I hope that this work reflects the positive impact you have had on me.

Acknowledgement

First and foremost, we would like to thank Almighty Allah for granting us the courage and patience to successfully complete this work. We would like to express our sincere gratitude to our advisors, **Dr.RAHIM** and **Dr.Lalouani** for their invaluable guidance and support throughout our master's program. their expertise, encouragement, and critical feedback have been extremely valuable in structuring the work and enhancing the quality of the various sections. We would like to extend our sincere appreciation to the members of the jury **Dr.RAFAA** and **DR.SAHMADI** for the honor they bestow upon us by taking the time to read and evaluate this work.

Our dear parents are our pride and the source of our success, as they have made sacrifices to provide us with a conducive working environment and the best possible conditions. Without them, nothing would have been easy. May God protect and bless them so that we can repay them for all that they have given us. We would also like to thank our friends and family for their love and support during this process. Without their encouragement and motivation, we would not have been able to complete this journey

Abstract

This work focuses on the design and development of a medical assistant application based on large language models. The goal is to create an intelligent system that can assist healthcare professionals and patients by leveraging advanced natural language processing capabilities.

The work explores the use of state-of-the-art language models to power the medical assistant, enabling it to engage in natural conversations, answer queries, and provide personalized recommendations related to healthcare. Key aspects covered include the system architecture, data processing techniques, and the integration of the language model with domain-specific medical knowledge.

Through rigorous testing and evaluation, including assessments on real-world clinical scenarios, the work aims to demonstrate the effectiveness of the medical assistant in enhancing healthcare delivery by applying evaluation techniques.

Keywords: Medical assistant, Large language models, Natural language processing, Healthcare, Chatbot, Data processing, Domain-specific knowledge, AI-powered healthcare, Diagnosis

Résumé

Ce travail se concentre sur la conception et le développement d'une application d'assistant médical basée sur des modèles de langage de grande taille. L'objectif est de créer un système intelligent capable d'assister les professionnels de la santé et les patients en exploitant les capacités avancées de traitement du langage naturel.

Le travail explore l'utilisation de modèles de langage de pointe pour alimenter l'assistant médical, lui permettant de s'engager dans des conversations naturelles, de répondre à des requêtes et de fournir des recommandations personnalisées liées aux soins de santé. Les aspects clés couverts incluent l'architecture du système, les techniques de traitement des données et l'intégration du modèle de langage avec des connaissances médicales spécifiques au domaine.

Grâce à des tests et évaluations rigoureux, y compris des évaluations sur des scénarios cliniques réels, ce travail vise à démontrer l'efficacité de l'assistant médical dans l'amélioration de la prestation des soins de santé en appliquant des techniques d'évaluation.

Mots-clés : Assistant médical, Modèles de langage de grande taille, Traitement du langage naturel, Soins de santé, Chatbot, Traitement des données, Connaissances spécifiques au domaine, Soins de santé alimentés par l'IA, Diagnostic

ملخص

هذا العمل يركز على تصميم وتطوير تطبيق مساعد طبي يعتمد على نماذج اللغة الكبيرة. الهدف هو إنشاء نظام ذكي يمكنه مساعدة المتخصصين في الرعاية الصحية والمرضى من خلال الاستفادة من قدرات معالجة اللغة الطبيعية المتقدمة.

يستكشف العمل استخدام نماذج اللغة الحديثة لتشغيل المساعد الطبي، مما يمكنه من إجراء محادثات طبيعية، والإجابة على الاستفسارات، وتقديم توصيات مخصصة تتعلق بالرعاية الصحية. تشمل الجوانب الرئيسية التي يغطيها العمل هيكل النظام وتقنيات معالجة البيانات ودمج نموذج اللغة مع المعرفة الطبية الخاصة بالمجال.

من خلال الاختبار والتقييم الدقيقين، بما في ذلك التقييمات في سيناريوهات سريرية واقعية، يهدف العمل إلى إظهار فعالية المساعد الطبي في تحسين تقديم الرعاية الصحية من خلال تطبيق تقنيات التقييم.

الكلمات المفتاحية: مساعد طبي، نماذج اللغة الكبيرة، معالجة اللغة الطبيعية، الرعاية الصحية، شات بوت، معالجة البيانات، المعرفة الخاصة بالمجال، الرعاية الصحية المعتمدة على الذكاء الاصطناعي، التشخيص

List of Acronyms

AI *Artificial Intelligence*

API *Application Programming Interface*

CDSS *Clinical Decision Support System*

CNN *Convolutional Neural Network*

DL *Deep Learning*

EHR *Electronic Health Record*

EMR *Electronic Medical Record*

FHIR *Fast Healthcare Interoperability Resources*

GPT *Generative Pre-trained Transformer*

LLM *Large Language Model*

ML *Machine Learning*

mHealth *Mobile Health*

NLP *Natural Language Processing*

RAG *Retrieval-Augmented Generation*

RNN *Recurrent Neural Network*

SVM *Support Vector Machine*

UI *User Interface*

Contents

Introduction **1**

1 Large Language Models (LLMs) **5**

1.1 Introduction 5

1.2 Concept of Large Language Models 6

1.2.1 History 6

1.2.2 Definition 8

1.2.3 Large Language Models Principles 9

1.2.4 Capabilities of Large Language Models 12

1.3 Applications of Large Language Models 13

1.3.1 Language Comprehension and Generation Tasks 13

1.3.2 Chatbots and Virtual Assistants 13

1.3.3 Education and Research 14

1.3.4 Content Creation and Marketing 14

1.3.5 Software Development 14

1.4 Challenges and Considerations with Large Language Models 15

1.4.1 Bias and Fairness in Training Data 15

1.4.2 Explainability and Interpretability of LLM Outputs 15

1.4.3 Safety and Security Concerns 15

1.4.4 Ethical Considerations Regarding LLM Use 16

1.4.5 Limitations in preformances 16

1.5 Prompt Engineering 17

1.6 Retrieval-Augmented Generation (RAG) 17

1.6.1 Introduction to RAG 17

1.6.2 Definition 18

1.6.3 Components of RAG 18
1.6.4 Advantages of RAG 19
1.6.5 Embeddings for RAG 20
1.6.6 Sentence Transformers 20
1.7 Conclusion 22

2 AI applications in medical field 23
2.1 Introduction 23
2.2 Telemedicine Platforms 24
2.2.1 Video Conferencing Tools 24
2.2.2 Secure Messaging Platforms 24
2.2.3 Remote Monitoring Devices 24
2.2.4 Mobile Health (mHealth) Applications 27
2.3 Artificial Intelligence (AI) Algorithms for Medical Diagnosis 29
2.3.1 Machine Learning Algorithms: 30
2.3.2 Deep Learning Models: 30
2.4 LLMs applications in medical feilds 31
2.4.1 Clinical Decision Support 31
2.4.2 Medical Research and Knowledge Discovery 32
2.5 Conclusion 33

3 Theoretical Framework 35
3.1 Introduction 35
3.2 System Requirements 36
3.3 Overvall architecture 37
3.4 Details descriptions 41
3.4.1 Option 1: Native Diagnosis 42
3.4.2 Option 2: Assisted Diagnosis 44
3.4.3 Datasets 45
3.5 Evaluation 45
3.5.1 Semantic Similarity 46
3.5.2 LangChain String Evaluators 47
3.6 Conclusion 49

4 Operational framework **51**

4.1 Introduction 51

4.2 Development Environment 51

4.2.1 Python 52

4.3 Implementation Details 55

4.3.1 Chatbot Implementation 55

4.3.2 Symptom Checker API Integration 57

4.4 Graphical User interface 58

4.5 Evaluation 63

4.5.1 Results and Discussions 63

4.6 Conclusion 66

Conclusion **67**

References **68**

List of Figures

1.1 NLP to LLMs timeline 6
1.2 The Transformer - model architecture 11

2.1 smart Blood Pressure Monitor 25
2.2 Smart Glucometer 26
2.3 Wearable Fitness Tracker 26
2.4 Symptom Tracking 28
2.5 Medication Reminder 29

3.1 Use Case Diagram of system interactions. 39
3.2 Native Diagnosis Use Case Diagram 40
3.3 Assisted Diagnosis Use Case Diagram 41
3.4 Workflow Diagram of System Interactions case.N°1. 43
3.5 Workflow Diagram of System Interactions case.N°2. 44
3.6 Evaluation Overview 48

4.1 Langchain Modules 54
4.2 Login Interface 59
4.3 Method Selection Interface 60
4.4 Assisted Diagnosis Interface 61
4.5 Selecting symptoms Tags 61
4.6 Native Diagnosis Interface 62
4.7 Conversation Interface 63

General Introduction

Healthcare is a fundamental aspect of our daily lives, dedicated to enhancing well-being and improving overall quality of life. With the continuous advancement of technology, particularly in Artificial Intelligence (AI), the concept of personal assistants and chatbots has significantly evolved. This evolution is exemplified by the development of Language Models (LLMs) like ChatGPT, which reflect the growing maturity of AI. As these models expand their knowledge base through extensive internet exploration, they demonstrate an increasing ability to achieve human-level performance on medical question-and-answer benchmarks. This progress hints at the potential for integrating Conversational AI into healthcare services. However, it is crucial to recognize and address the inherent limitations of these models, such as their tendency to reflect biases, establish unwarranted associations, and produce inaccurate outputs when handling critical biomedical data.

The focus of this project is the development of a healthcare assistant designed to aid users in obtaining accurate diagnostics by analyzing symptoms associated with common diseases. Despite the promising advancements of LLMs in the medical domain, several key issues and challenges must be addressed for their successful integration into healthcare applications.

Firstly, data quality and availability are critical. LLMs require vast amounts of high-quality, diverse, and unbiased medical data for training. However, obtaining such data is challenging due to privacy and regulatory concerns, as well as the potential biases in existing datasets. Secondly, ensuring accuracy and reliability is paramount. While LLMs perform impressively on medical question-answering benchmarks, their outputs can still be inaccurate or unreliable, particularly in critical healthcare scenarios. Accuracy and reliability are essential for patient safety and trustworthiness.

Another significant challenge is explainability and interpretability. LLMs are often perceived as ”black boxes,” making it difficult to understand and interpret their decision-making processes. In healthcare, where transparency and accountability are crucial, improving the explainability and interpretability of LLM-based systems is essential. Finally, gaining user trust and adoption is a hurdle. Even with accurate and

reliable outputs, users may be hesitant to rely on AI for critical healthcare decisions.

The primary objectives of this study are to address these challenges systematically. First, identifying and acquiring relevant, high-quality medical data sources and preprocessing this data for effective LLM training is essential. Second, exploring and evaluating different LLM architectures and training strategies will optimize their performance for medical diagnosis tasks. Third, developing robust evaluation methodologies to assess the accuracy and reliability of the LLM-based medical assistant is crucial to ensure it meets the required standards for healthcare applications. Fourth, investigating techniques to improve the explainability and interpretability of the LLM-based medical assistant will enable users to understand the reasoning behind its outputs. Lastly, exploring user experience design principles and adoption strategies will enhance user trust and facilitate the seamless integration of the LLM-based medical assistant into healthcare workflows.

By addressing these objectives, this study aims to contribute to the development of accurate, reliable, and trustworthy LLM-based medical assistants. These advancements have the potential to significantly enhance healthcare services and improve patient outcomes.
our thesis outline structure as following :
chapter 1 This chapter will provide an overview of Large Language Models (LLMs), discussing how they leverage deep learning and transformer architectures to enable advanced natural language processing capabilities. It will explore the technical foundations of LLMs, their training methodologies, and the key characteristics that have led to their widespread adoption across various domains.

chapter 2 will focus on exploring the transformative applications of AI, particularly LLMs, within the medical domain. It will discuss into specific use cases such as Telemedicine platforms, mobile health applications,Development of medical chatbots and clinical assistance.The chapter also highlight how these AI-driven applications can contribute to improving healthcare delivery for human.

chapter 3 This chapter will provide a general overview of the approaches and methodologies for designing a medical assistant system based on Large Language Models

(LLMs).

chapter 4 present the implementation and evaluation of the proposed technic,This will include providing a practical steps of the various tools used to develop the medical assistant to be rely on LLMs.

Chapter 1

Large Language Models (LLMs)

1.1 Introduction

Large Language Models (LLMs) have emerged as one of the most transformative technologies in the field of AI specifically in Natural Language Processing (NLP), revolutionizing how machines understand and generate human language. With the exponential growth of digital data and computational power, LLMs have achieved remarkable milestones, showcasing their potential to redefine various aspects of communication, automation, and knowledge management[1].

Recent advancements in LLMs, particularly exemplified by models like OpenAI's GPT (Generative Pre-trained Transformer) series and Google's BERT (Bidirectional Encoder Representations from Transformers), have demonstrated unprecedented capabilities in tasks such as text generation, translation, summarization, and question answering[2]. For instance, OpenAI's GPT-3 model, with its 175 billion parameters, has showcased impressive abilities in generating coherent and contextually relevant text across diverse domains, surpassing previous benchmarks in NLP tasks.

The significance of LLMs lies not only in their performance but also in their potential applications across various domains. From aiding content creation and enhancing human-computer interaction to advancing information retrieval and knowledge extraction, LLMs hold the promise of reshaping industries, academia, and everyday life[3].

The objective of this chapter is to provide a comprehensive overview of LLMs, covering their evolution, core concepts, training methodologies, capabilities, limitations, and future directions. By delving into these aspects, we aim to offer insights into the workings of LLMs, their potential impact on society, and the challenges and opportunities they present in the realm of NLP and beyond.

1.2 Concept of Large Language Models

1.2.1 History

Natural Language Processing (NLP) has a rich history that traces back several decades, marked by significant milestones that have shaped its evolution into the sophisticated field it is today. The journey of NLP can be characterized by a series of breakthroughs in linguistic theory, computational algorithms, and the availability of large-scale datasets.the figure below show the main events in NLP until LLMs journey :

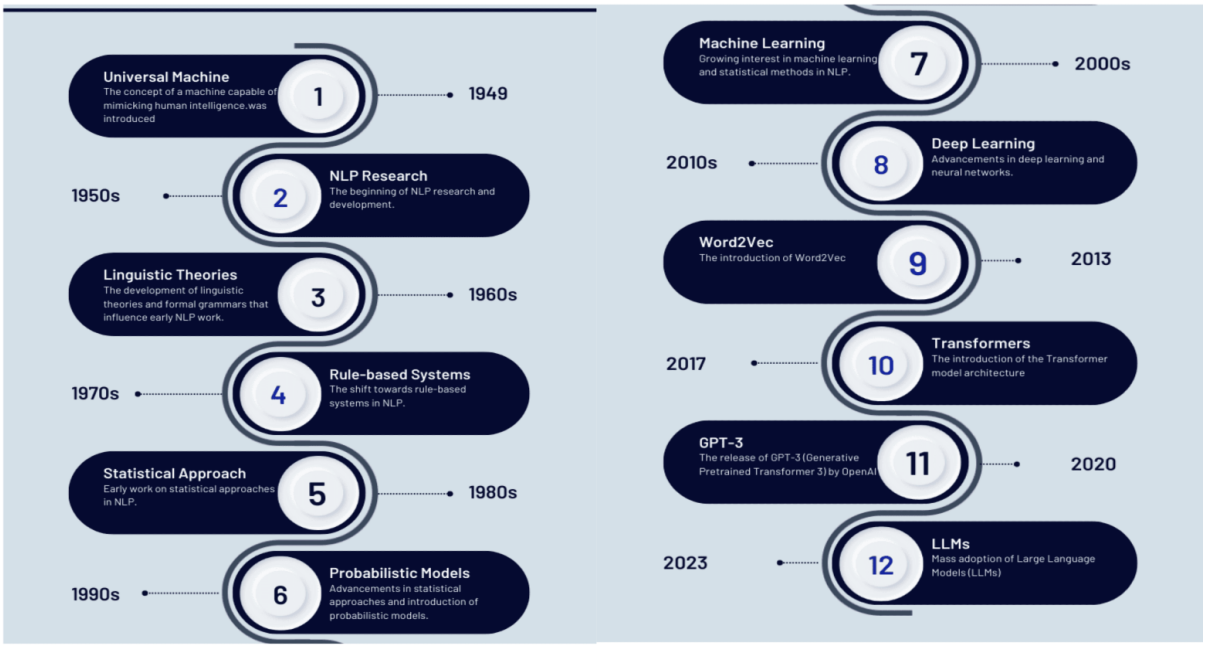


Figure 1.1: NLP to LLMs timeline

The concept of a "universal machine" capable of mimicking human intelligence was proposed by *Alan Turing* in 1949, laying the groundwork for future advancements in artificial intelligence and NLP. The earliest endeavors in NLP can be dated back to the 1950s, with the beginnings of NLP research and development. During this period,

rule-based systems were developed to analyze and understand human language, focusing on simple tasks such as language translation and information retrieval. One notable achievement from this era was the 1954 Georgetown-IBM experiment, which used an IBM 701 computer for Russian-English translation, marking one of the earliest machine translation experiments.

The development of linguistic theories and formal grammar in the 1960s further influenced early NLP work. This period saw the introduction of various linguistic models that aimed to formalize the structure of human language. The 1970s witnessed a shift towards rule-based systems in NLP, with notable advancements such as Terry Winograd's SHRDLU in 1972, an effective NLP system that could manipulate blocks in a virtual world using natural language commands.

One of the key milestones in the history of NLP was the introduction of statistical methods in the 1980s. Early work on statistical approaches, such as the development of the Hidden Markov Model Toolkit (HTK) in 1989, helped researchers build statistical models for speech recognition. The 1990s saw continued advancements in statistical approaches and the introduction of probabilistic models such as probabilistic context-free grammar (PCFG). Techniques like n-grams, hidden Markov models, and maximum entropy models revolutionized how language data could be processed and analyzed. However, these traditional NLP techniques had inherent limitations, such as the need for handcrafted features and the inability to capture complex linguistic patterns effectively.

The 2000s marked a growing interest in machine learning and statistical methods in NLP. The emergence of neural networks, particularly recurrent neural networks (RNNs), marked another significant advancement in NLP. RNNs, with their ability to process sequential data, showed promise in tasks like language modeling and machine translation. However, they struggled with long-range dependencies and were computationally expensive to train, limiting their effectiveness in handling large-scale language data.

The turning point in NLP came with the rise of Large Language Models (LLMs), powered by transformer architectures. The 2010s saw a resurgence of interest in NLP

driven by advancements in deep learning and neural networks. In 2013, the introduction of Word2Vec, a word embedding technique that represents words as dense vectors, significantly improved NLP models' performance. The development of Google's neural network-based machine translation system, Google Neural Machine Translation (GNMT) in 2014, also marked a significant improvement in translation quality.

Transformers, introduced in the seminal paper *"Attention is All You Need"* by Vaswani et al. (2017) [4], revolutionized how information could be processed in neural networks, enabling more efficient handling of long-range dependencies and parallel computation. LLMs, such as OpenAI's GPT series and Google's BERT, built upon the transformer architecture, achieving unprecedented performance across a wide range of NLP tasks. In 2020, the release of GPT-3 (Generative Pretrained Transformer 3) by OpenAI, one of the most significant language models to date, showcased the capability to generate coherent and contextually relevant text. The advancements in 2021 in zero-shot and few-shot learning further enabled models to perform well on tasks without extensive task-specific training data.

LLMs address many of the limitations of traditional NLP techniques by leveraging vast amounts of unlabeled text data for pre-training, enabling them to learn rich representations of language without the need for handcrafted features. Additionally, the self-attention mechanism in transformer models allows LLMs to capture contextual information effectively, overcoming the shortcomings of RNNs in modeling long-range dependencies.

1.2.2 Definition

Large Language Models (LLMs) are a class of natural language processing (NLP) models characterized by their massive parameter size, extensive training on vast amounts of text data, and proficiency in handling complex language tasks[2]. These models have revolutionized the field of NLP by achieving state-of-the-art performance across various language understanding and generation tasks.

The key characteristic of LLMs is their massive parameter size, which enables them to capture intricate linguistic patterns and nuances present in natural language text.

For example, models such as OpenAI's GPT (Generative Pre-trained Transformer) series and Google's BERT (Bidirectional Encoder Representations from Transformers) have billions of parameters, allowing them to encode and process large amounts of textual information[3].

Training LLMs requires substantial computational resources and vast amounts of text data[1]. These models are typically pre-trained on large-scale corpora, such as web text, books, and articles, using unsupervised learning techniques. During pre-training, LLMs learn to predict the next word in a sequence given the context of preceding words, which helps them capture semantic and syntactic structures inherent in language[5].

Moreover, LLMs leverage neural network architectures as their underlying foundation. Neural networks consist of interconnected nodes organized in layers, where each node represents a mathematical operation applied to the input data[6]. The architecture of LLMs, particularly transformer-based architectures, plays a crucial role in their ability to process sequential data efficiently.

Overall, LLMs represent a significant advancement in NLP, offering unparalleled capabilities in understanding and generating natural language text. Their massive parameter size, extensive training data, and neural network-based architecture enable them to excel in a wide range of language tasks, from text generation to sentiment analysis.

1.2.3 Large Language Models Principles

Large Language Models (LLMs) operate by leveraging extensive training data to learn the complex patterns and structures inherent in natural language. The process involves several key components and techniques:

Training data plays a role in the development of LLMs, providing the models with the necessary linguistic knowledge and context. LLMs are typically trained on vast amounts of text data, sourced from diverse corpora such as books, articles, websites, and social media posts. The abundance and diversity of training data are crucial for enabling LLMs to generalize well across various language tasks and domains[1].

LLMs can be trained using both supervised and unsupervised learning techniques. In supervised learning, the model learns from labeled data, where each input is paired with a corresponding target output. This approach is commonly used for tasks like sentiment analysis and text classification, where the model is trained to predict predefined labels or categories[7]. On the other hand, unsupervised learning involves training the model on unlabeled data, allowing it to learn patterns and structures from the input data without explicit supervision. Techniques such as self-supervised learning, where the model learns from the data itself by predicting masked or corrupted tokens, have been particularly effective in training LLMs[2].

Transformers have become as a dominant architecture for LLMs due to their ability to capture long-range dependencies and contextual information in text. The transformer architecture, introduced by Vaswani et al.[4] as shown in the figure1.2 below, employs self-attention mechanisms that enable the model to weigh the relevance of different words in a sequence, facilitating better contextual understanding[8]. This attention mechanism allows LLMs to efficiently process input sequences of varying lengths and capture dependencies across distant tokens, overcoming the limitations of traditional recurrent neural networks (RNNs) and convolutional neural networks (CNNs)[4]. Transformers have been widely adopted in state-of-the-art LLMs such as BERT[2], GPT[3], and RoBERTa [9], contributing to their remarkable performance across diverse language tasks.

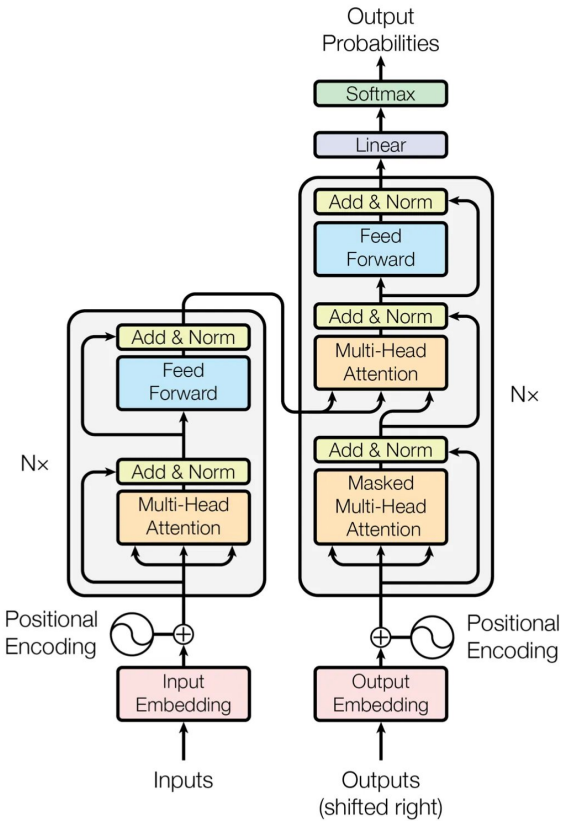


Figure 1.2: The Transformer - model architecture

[4]

In summary, LLMs leverage extensive training data, supervised and unsupervised learning techniques, and transformer-based architectures to achieve state-of-the-art performance in natural language processing tasks.

1.2.4 Capabilities of Large Language Models

Large Language Models (LLMs) have demonstrated remarkable proficiency across various natural language processing (NLP) tasks. Below are specific examples showcasing their capabilities:

a. Text Generation

LLMs exhibit impressive proficiency in generating coherent and contextually relevant text across different genres and styles. They can produce creative content, including poetry, stories, and news articles. For instance, models like GPT-3 have shown the ability to generate human-like text with minimal prompts.[3]

b. Machine Translation

LLMs have significantly advanced machine translation systems by generating accurate translations across multiple languages. Models such as BERT and T5 have achieved state-of-the-art performance in translation tasks, surpassing traditional statistical approaches[2].

c. Text Summarization

LLMs excel in generating concise and informative summaries of longer texts. By leveraging attention mechanisms and transformer architectures, models like BART and PEGASUS can effectively distill key information from documents, articles, and news stories[10].

d. Question Answering

LLMs have been instrumental in developing question answering systems capable of understanding and responding to natural language queries. Models like BERT and RoBERTa achieve high accuracy in answering factual questions based on context[9].

e. Code Generation

LLMs have shown promise in generating code snippets and assisting developers in programming tasks. By training on large code repositories, models like Codex and

GPT-3 can generate syntactically correct code for various programming languages based on natural language descriptions[11].

These examples underscore the versatility and effectiveness of LLMs in addressing diverse language processing tasks.

1.3 Applications of Large Language Models

Large Language Models (LLMs) have garnered significant attention due to their wide-ranging applications across various domains. Their ability to understand and generate human-like text has opened up numerous possibilities for their use in diverse fields:

1.3.1 Language Comprehension and Generation Tasks

LLMs serve as the backbone for numerous NLP applications, including sentiment analysis, named entity recognition, and part-of-speech tagging. Models like BERT and GPT have been extensively used in research and industry to achieve state-of-the-art results in various NLP benchmarks[2]. These models excel in understanding context and disambiguating meanings, which are critical for accurate language comprehension and generation. For instance, BERT's bidirectional training allows it to capture the context from both directions, enhancing its performance in tasks such as question answering and text summarization[9]. GPT, with its autoregressive nature, excels in generating coherent and contextually relevant text, making it suitable for tasks like story generation and dialogue systems[3].

1.3.2 Chatbots and Virtual Assistants

LLMs power chatbots and virtual assistants by enabling them to understand and generate human-like responses in conversations. Systems like OpenAI's GPT-based chatbots have been integrated into customer service platforms, social media platforms, and virtual assistants to provide personalized interactions and assistance[12]. These systems leverage the models' ability to maintain context over multiple turns of conversation, making interactions more natural and engaging. Furthermore, advancements in fine-tuning techniques allow these models to adapt to specific domains, en-

hancing their effectiveness in specialized applications such as mental healthcare and finance[13].

1.3.3 Education and Research

In the field of education, LLMs facilitate intelligent tutoring systems, automated essay grading, and content recommendation systems. Researchers leverage LLMs to analyze educational texts, generate study materials, and develop adaptive learning platforms[14]. For example, automated grading systems can provide instant feedback on student essays, helping educators to scale their assessments while maintaining consistency[15]. Additionally, LLMs can generate personalized learning materials based on individual student needs, thereby promoting adaptive learning and improving educational outcomes[16].

1.3.4 Content Creation and Marketing

LLMs are employed in content generation tasks such as writing articles, product descriptions, ad copies, and social media posts. By understanding user preferences and generating engaging content, LLMs aid marketers in content creation, audience engagement, and brand promotion[17]. These models can analyze vast amounts of data to identify trending topics and optimize content for target audiences. Moreover, the ability to generate high-quality text rapidly allows marketers to scale their content production efforts, thereby enhancing their online presence and outreach[18].

1.3.5 Software Development

LLMs assist software developers in various tasks, including code generation, code summarization, and bug detection. Models like GPT-3 and Codex can generate code snippets based on natural language descriptions, speeding up development processes and enhancing code quality[11]. These models can also help in understanding and documenting existing codebases, making it easier for developers to maintain and extend software projects[19]. Furthermore, LLMs can suggest improvements and detect potential errors in code, thereby contributing to more robust and reliable software[20].

These applications highlight the versatility and effectiveness of LLMs in transforming various sectors through natural language understanding and generation capabilities. As these models continue to evolve, their potential to innovate and improve processes across different domains is expected to grow even further.

1.4 Challenges and Considerations with Large Language Models

Large Language Models (LLMs) offer remarkable capabilities, but their widespread adoption also raises several challenges and considerations:

1.4.1 Bias and Fairness in Training Data

One significant challenge is the presence of biases in the training data, which can perpetuate and exacerbate societal biases in LLM outputs. Research has shown that LLMs trained on biased datasets can produce discriminatory or unfair results across various demographic groups[21]. Addressing bias in training data and developing algorithms for debiasing LLMs are crucial steps to ensure fair and equitable outcomes[22].

1.4.2 Explainability and Interpretability of LLM Outputs

LLMs, particularly deep neural networks, often operate as black boxes, making it challenging to understand how they arrive at their decisions. Lack of explainability and interpretability raises concerns about trust and accountability, especially in critical applications such as healthcare and law[23]. Research efforts focus on developing techniques to interpret and explain LLM outputs, enhancing transparency and trust in their decision-making processes[24].

1.4.3 Safety and Security Concerns

LLMs have the potential to generate harmful or misleading content, including fake news, hate speech, and misinformation. Malicious actors could exploit LLMs to manipulate public opinion or launch sophisticated phishing attacks[25]. Mitigating safety

and security risks requires robust mechanisms for content moderation, detection of malicious intent, and proactive measures to prevent misuse of LLMs[26]

1.4.4 Ethical Considerations Regarding LLM Use

The widespread deployment of LLMs raises complex ethical dilemmas, including issues of privacy, consent, and autonomy. Ethical concerns also extend to the environmental impact of training large models, as it consumes significant computational resources and energy[27].Ethical frameworks and guidelines are essential to guide the responsible development and deployment of LLMs, ensuring they align with societal values and ethical principles[28]

1.4.5 Limitations in preformances

Despite the fascination with the magical capabilities of Large Language Models (LLMs) in generating text and performing various language-related tasks, it is essential to acknowledge the struggles and limitations in their performances[29]. In addition to these limitations, LLMs can exhibit phenomenon hallucinations, which refers to the generation of text that is not based on factual information or accurate understanding of the context. Let's explore some of the limitations of LLMs and how they can contribute to hallucinations:

a. Lack of Contextual Understanding: LLMs primarily rely on statistical patterns in the training data to generate text. While they can generate syntactically correct sentences, they often lack a deep understanding of the context and meaning behind the text they produce. This can lead to inaccurate or nonsensical responses[29].

b. Limited Knowledge Base: LLMs are trained on vast amounts of data, but they may still lack comprehensive knowledge on specific topics. They may struggle to provide accurate information or answer complex questions outside the scope of their training data. This limitation can contribute to hallucinations, as the model may generate responses that are not grounded in factual information[29].

c. Difficulty with Ambiguity and Nuance: LLMs often struggle with understanding and interpreting ambiguous or nuanced language. They may provide simplistic or incorrect interpretations of complex statements or questions, leading to mislead-

ing or inaccurate responses. Hallucinations can arise when the model fills in gaps in understanding with fabricated or incorrect information[30].

Addressing these challenges and considerations is paramount to harnessing the full potential of LLMs while mitigating potential risks and ensuring ethical and equitable use.

1.5 Prompt Engineering

Prompt engineering refers to the process of crafting effective prompts or input text to elicit desired outputs from large language models (LLMs)[31]. Prompt engineering involves carefully structuring the prompt to guide the LLM towards generating coherent, relevant, and high-quality responses. This can include techniques such as:

- Specifying the task or objective clearly in the prompt
- Providing relevant context, background information, or constraints
- Using formatting, keywords, or special tokens to signal the LLM
- Breaking down complex tasks into step-by-step instructions
- Iteratively refining the prompt based on the model's outputs

Effective prompt engineering is critical for using the capabilities of modern LLMs to solve a wide variety of natural language processing tasks. It allows users to harness the knowledge and generation abilities of these large neural networks in a controlled and customized manner[32][33].

1.6 Retrieval-Augmented Generation (RAG)

1.6.1 Introduction to RAG

The concept of Retrieval-Augmented Generation (RAG) represents a paradigm shift in natural language processing (NLP), where traditional language generation models are augmented with a retrieval mechanism to enhance the relevance and coherence of generated text[34]. RAG systems integrate the strengths of both generative and

retrieval-based approaches, enabling the generation of text that is not only fluent but also grounded in real-world knowledge[35].

1.6.2 Definition

At the core of RAG lies the idea of combining the fluency of generative models, such as GPT (Generative Pre-trained Transformer)[3], with the factual accuracy and relevance provided by retrieval mechanisms. In RAG systems, the retrieval component retrieves relevant context or knowledge from a large external database or corpus, which is then used to guide the generation process[36]. This integration allows RAG models to produce text that is not only grammatically correct but also contextually appropriate and factually accurate[37].

1.6.3 Components of RAG

The components of RAG can be viewed in two main aspects: the Retrieval Mechanism and the Generation Mechanism:

- **The Retrieval Mechanism** in RAG systems serves as the foundation for incorporating external knowledge into the generation process. This component retrieves relevant information or context from a large knowledge source, such as a text corpus or a knowledge graph, to guide the generation of coherent and contextually relevant text[34]. Various retrieval techniques have been employed in RAG systems, ranging from simple keyword matching to more sophisticated information retrieval methods, such as dense retrievers based on neural networks[38]. These retrieval mechanisms aim to select the most informative and contextually relevant knowledge for the generation task at hand, thereby enhancing the quality and coherence of the generated text.

- **The Generation Mechanism** in RAG systems is responsible for producing text based on the retrieved information and the input provided to the model[3]. This component typically involves the use of large-scale language models, such as GPT (Generative Pre-trained Transformer), which generate text autoregressively based on the retrieved context and the model's learned parameters[4]. The generation mechanism in RAG systems is trained to produce text that is not only

grammatically correct but also coherent and contextually relevant to the retrieved knowledge[37]. By integrating the retrieval and generation mechanisms, RAG systems are able to leverage both external knowledge and learned language patterns to generate high-quality text across various NLP tasks.

1.6.4 Advantages of RAG

Improved Coherence and Relevance in Generated Text

Retrieval-Augmented Generation (RAG) significantly enhances the coherence and relevance of generated text by leveraging external knowledge sources. Traditional language models generate responses based solely on the information embedded in their training data, which can lead to limitations in accuracy and contextual relevance. In contrast, RAG combines retrieval mechanisms with generative capabilities, allowing the model to fetch pertinent information from large-scale databases or the web in real-time before generating responses[34]. This process ensures that the generated content is not only contextually appropriate but also factually accurate and up-to-date.

Research indicates that incorporating a retrieval step improves the model's ability to maintain topic consistency over extended interactions, which is particularly beneficial in applications like customer service and interactive dialogue systems[39]. For instance, a RAG model can retrieve specific product details or policy information during a customer support interaction, thereby providing precise and reliable answers that enhance user satisfaction[38].

Mitigating Biases Through Informed Generation

Another significant advantage of RAG is its potential to mitigate biases present in language models. Traditional language models are often trained on large datasets that inadvertently contain various biases, which can lead to the generation of biased or harmful content. By integrating a retrieval mechanism, RAG can access diverse and more balanced external knowledge bases, which helps in counteracting these biases[40].

Studies have demonstrated that the retrieval component in RAG can be designed to prioritize sources that are known for their balanced and fair perspectives, thus providing a counterbalance to the biases embedded in the training data[41]. For ex-

ample, when generating content related to socio-political topics, a RAG model can retrieve information from reputable and diverse sources, thereby presenting a more nuanced and equitable view[42].

Additionally, the retrieval process can be dynamically adjusted based on the context of the query, ensuring that the most relevant and unbiased information is used during generation. This adaptability is crucial in producing content that is both contextually appropriate and ethically sound[43][44].

1.6.5 Embeddings for RAG

Embeddings are a fundamental component in the RAG architecture, serving as a bridge between the retrieval and generation processes. They transform textual data into dense vector representations, which can be efficiently compared and processed by machine learning algorithms. These vector representations capture semantic meanings, allowing the model to identify and retrieve relevant passages from a vast knowledge base[5].

In the context of RAG, embeddings enable the model to perform similarity searches between the user's query and potential information sources. When a query is input into the system, it is encoded into an embedding vector. The retrieval mechanism then searches for passages with similar embeddings, ensuring that the most contextually relevant information is retrieved to augment the language model's generation[38]. This process enhances the relevance and coherence of the generated responses by grounding them in factual and contextually appropriate information.

1.6.6 Sentence Transformers

Sentence Transformers are type of deep learning model that encode text or phrase into high-dimensional vector representations,the model what apply sentence embeddings which means its nummeric representation of sentences.

Sentence transformers main role is encoding both queries and passages in the RAG framework. These models, based on transformer architectures, are designed to generate embeddings for entire sentences, capturing the contextual nuances and semantic relationships within the text[45].

In a RAG system, sentence transformers encode queries and candidate passages into dense vectors in a shared embedding space. This shared space allows for efficient and accurate similarity comparisons, facilitating the retrieval of the most relevant passages to support the generative process[46]. By leveraging sentence transformers, RAG models can handle complex queries and retrieve contextually rich information, thereby improving the quality and relevance of the generated text[47].

1.7 Conclusion

Large Language Models (LLMs) represent a significant advancement in natural language processing, offering unprecedented capabilities and opening up a wide range of potential applications. Throughout this chapter, we have explored the key aspects of LLMs, their capabilities, and potential future directions.

LLMs have demonstrated remarkable proficiency across various language tasks, including text generation, machine translation, summarization, question answering, and code generation. They leverage massive parameter sizes and vast amounts of training data to understand and generate human-like text with impressive accuracy.

The potential applications of LLMs span across numerous domains, including but not limited to natural language processing tasks, chatbots and virtual assistants, education and research, content creation and marketing, and software development. These models have the capability to automate tasks, enhance productivity, and revolutionize how we interact with and process textual data.

Looking ahead, the future of LLM development holds exciting prospects. Advancements in model architectures, training techniques, and ethical guidelines will shape the evolution of these models. Researchers are exploring techniques to improve model efficiency, scalability, and interpretability while addressing challenges such as bias, fairness, safety, and ethical considerations.

Furthermore, the integration of multimodal capabilities, combining text with other modalities like images and audio, presents new avenues for LLMs to better understand and generate content. Additionally, the development of more robust pre-training methods and fine-tuning strategies will continue to enhance the performance and adaptability of LLMs to specific tasks and domains.

In conclusion, LLMs have emerged as powerful tools in natural language processing, with the potential to revolutionize various fields and industries. While significant progress has been made, ongoing research and development efforts are essential to address challenges and unlock the full potential of these models in the years to come.

Chapter 2

AI applications in medical field

2.1 Introduction

Artificial Intelligence (AI) has brought significant advancements across various industries, with the medical field experiencing some of the most transformative changes. Among the array of AI technologies, Large Language Models (LLMs) have emerged as pivotal tools in enhancing medical practice. These models, including GPT-4, leverage vast amounts of data and sophisticated algorithms to process and generate human-like text, thereby offering numerous applications in healthcare.

The purpose of this chapter is to explore the multifaceted applications of AI in the medical field, with a particular focus on LLMs. We will examine how these models and other AI-driven technologies are reshaping telemedicine platforms, aiding in medical diagnosis, and supporting clinical decision-making[48]. By analyzing specific use cases such as telemedicine, mobile health applications, machine learning algorithms, and the role of LLMs in clinical and research settings, this chapter aims to provide a comprehensive understanding of the current and potential impacts of AI on healthcare.

From facilitating remote consultations and real-time patient monitoring to enhancing the accuracy of medical diagnoses and personalizing treatment plans, AI is revolutionizing healthcare delivery. This chapter will delve into how these technologies improve patient outcomes, streamline clinical workflows, and foster advancements in medical research, thereby contributing to more efficient and effective healthcare systems.

2.2 Telemedicine Platforms

Telemedicine platforms leverage telecommunications technology to deliver healthcare remotely, enabling patients to access medical assistance from a distance. These platforms encompass a variety of tools and technologies, including video conferencing, secure messaging and remote monitoring devices[49]. lets shed light on a few of them:

2.2.1 Video Conferencing Tools

Video conferencing tools facilitate real-time communication between patients and healthcare providers, allowing for virtual consultations and examinations. Platforms such as Zoom, Skype for Business, and Cisco Webex have been widely adopted in telemedicine due to their ease of use and robust features. Research has shown that video consultations can effectively support various medical specialties, including primary care, mental health, and dermatology[50].

2.2.2 Secure Messaging Platforms

Secure messaging platforms enable asynchronous communication between patients and healthcare providers, allowing for the exchange of text-based messages, images, and documents. Examples include platforms like Spruce Health and Klara, which offer HIPAA-compliant messaging solutions tailored for healthcare. Studies have demonstrated the effectiveness of secure messaging in improving patient engagement, satisfaction, and clinical outcomes[51].

2.2.3 Remote Monitoring Devices

Remote monitoring devices play a essential role in modern healthcare by allowing the collection of patient data remotely and providing real-time health monitoring and feedback. These devices, which include wearable fitness trackers, blood pressure monitors, glucometers, and various sensors, enable continuous monitoring of vital signs and health parameters outside of traditional clinical settings. Integration of these devices into e-medical assistant systems enhances patient care by providing

healthcare providers with timely and accurate data for informed decision-making and personalized interventions[52].

Utilization of Blood Pressure Monitors

Blood pressure monitors are essential for managing hypertension and monitoring cardiovascular health. With advancements in technology, portable blood pressure monitors that can be worn or used at home have become increasingly available[53]. Integration of these devices into e-medical assistant systems enables continuous monitoring of blood pressure levels, facilitating early detection of hypertension and timely adjustments to treatment plans. Healthcare providers can remotely access blood pressure data, track trends over time, and intervene promptly in case of abnormalities or fluctuations(see Figure 2.1).



Figure 2.1: smart Blood Pressure Monitor

Incorporation of Glucometers for Diabetes Management

Glucometers are vital tools for individuals with diabetes to monitor blood glucose levels regularly. Portable glucometers allow patients to perform self-monitoring of blood glucose (SMBG) at home or on the go[54]. By integrating glucometers into e-medical assistant systems, healthcare providers can remotely monitor patients' blood sugar levels, track glycemic control, and provide personalized guidance on medication dosing, dietary adjustments, and lifestyle modifications. Timely access to glucose data enables early detection of hypo- or hyperglycemic episodes and facilitates proactive interventions to prevent complications.



Figure 2.2: Smart Glucometer

Integration of Wearable Fitness Trackers

Wearable fitness trackers have gained popularity for their ability to monitor physical activity, sleep patterns, and other health-related metrics. These devices typically use sensors such as accelerometers, gyroscopes, and heart rate monitors to collect data[55]. By integrating wearable fitness trackers into e-medical assistant systems, healthcare providers can access real-time information on patients' activity levels, sleep quality, and exercise habits. This data can be used to assess overall health status, track progress towards fitness goals, and identify potential issues such as sedentary behavior or irregular sleep patterns.



Figure 2.3: Wearable Fitness Tracker

Incorporating remote monitoring devices into e-medical assistant systems enhances the delivery of healthcare services by enabling real-time health monitoring and feedback. Wearable fitness trackers, blood pressure monitors, and glucometers offer valuable insights into patients' physical activity, cardiovascular health, and glycemic control, respectively. By leveraging these devices, healthcare providers can remotely assess patients' health status, track progress, and intervene promptly to optimize outcomes.

2.2.4 Mobile Health (mHealth) Applications

Mobile health (mHealth) applications have gained significant attention in recent years due to their potential to improve healthcare accessibility, efficiency, and patient engagement. These apps, designed for smartphones and other mobile devices, offer a diverse range of functionalities aimed at enhancing healthcare delivery, including symptom tracking, medication management.here a close look at these applications:

- **Symptom Tracking:**

  mHealth apps enable users to monitor and track their symptoms, providing valuable data for both patients and healthcare providers. These apps often incorporate features such as symptom diaries, self-assessment tools, and personalized health journals. For example, the study by Dennison et al.[56] highlights the effectiveness of mHealth apps in empowering patients to track symptoms related to chronic conditions like diabetes and hypertension.

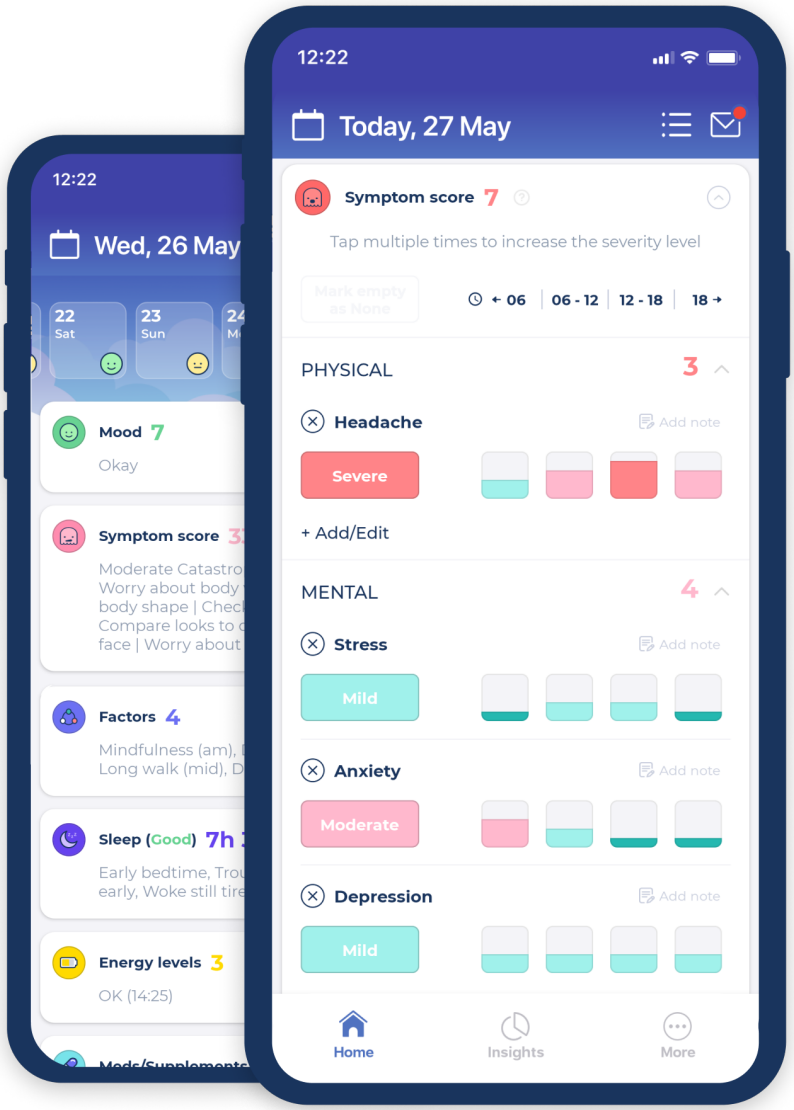


Figure 2.4: Symptom Tracking

- **Medication Reminders:**

  importance to medication regimens is a critical aspect of managing chronic diseases and ensuring optimal treatment outcomes. mHealth apps offer medication reminder functionalities, sending timely notifications and alerts to users to take their medications as prescribed. Research by Thakkar et al.[57]demonstrates the efficacy of medication reminder apps in improving medication adherence among patients with chronic conditions.

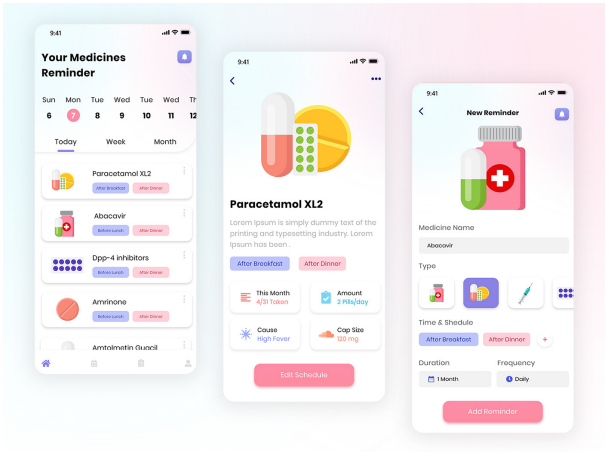


Figure 2.5: Medication Reminder

- **Health Behavior Tracking:**

  Beyond symptom monitoring and medication management, mHealth apps also support tracking of health behaviors such as physical activity, diet, and sleep patterns. By collecting and analyzing user-generated data, these apps can provide insights into lifestyle factors that impact overall health and wellness. research by Direito et al.[58] examines the role of mHealth apps in promoting physical activity and healthy behavior changes among users.

as a conclusion,Telemedicine platforms expanding access to healthcare services, particularly in underserved areas and during public health emergencies. By leveraging telecommunications technology, these platforms enable convenient and timely delivery of medical assistance, improving patient outcomes and reducing healthcare costs[59].

2.3 Artificial Intelligence (AI) Algorithms for Medical Diagnosis

Artificial intelligence (AI) algorithms trained on medical data have revolutionized medical diagnosis by enabling the analysis of vast amounts of patient data to assist healthcare providers in diagnosing conditions, predicting outcomes, and recommending treatment plans[60].

2.3.1 Machine Learning Algorithms:

Machine learning algorithms are commonly used in medical diagnosis due to their ability to learn patterns and relationships from data. Support Vector Machines (SVMs), Random Forests, and Neural Networks are among the most widely used machine learning algorithms in healthcare[61].

Support Vector Machines (SVMs):

SVMs are supervised learning models used for classification and regression analysis. They have been successfully applied in medical image analysis, such as tumor detection and classification in radiology[62].

Random Forests:

Random Forests are an ensemble learning method that operates by constructing a multitude of decision trees during training and outputting the mode of the classes (classification) or mean prediction (regression) of the individual trees. They have been used in various medical applications, including predicting disease risk and diagnosing conditions based on clinical data[63].

Neural Networks:

Neural networks, particularly deep learning architectures such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have shown remarkable performance in medical image analysis, natural language processing of clinical text, and predictive modeling of patient outcomes[64].

2.3.2 Deep Learning Models:

Deep learning models, a subset of machine learning algorithms, have gained popularity in medical diagnosis due to their ability to automatically learn features from raw data without manual feature engineering. These models are particularly effective in tasks such as medical image analysis and electronic health record (EHR) data processing[65].

Convolutional Neural Networks (CNNs):

CNNs have been extensively used for medical image analysis, including the detection and classification of abnormalities in radiology images such as X-rays, MRIs, and CT scans[66].

Recurrent Neural Networks (RNNs):

RNNs are well-suited for sequential data processing and have been applied in tasks such as predicting patient outcomes based on time-series clinical data and analyzing electronic health records[67].

Integration into e-Medical Assistant Systems:

These AI algorithms can be integrated into e-medical assistant systems to provide decision support for healthcare providers, assist in diagnosing conditions, and recommend personalized treatment plans based on patient data[60]. By leveraging the capabilities of machine learning and deep learning models, e-medical assistant systems can enhance the efficiency and accuracy of medical diagnosis in remote healthcare settings

2.4 LLMs applications in medical feilds

2.4.1 Clinical Decision Support

Enhancing Accuracy and Speed of Diagnosis Large Language Models like GPT-4 reasoning has improve diagnostic processes. By analyzing vast amounts of medical literature and patient data, LLMs can provide clinicians with accurate and timely diagnostic suggestions. For example, LLMs can aid in identifying rare diseases by matching patient symptoms with extensive medical databases, thereby reducing misdiagnosis and speeding up the diagnostic process[68]. Studies have shown that integrating LLMs into clinical workflows can score better diagnostic accuracy, particularly in complex cases where traditional methods may stagnate [69].

Supporting Differential Diagnosis Processes where LLMs can offer a list of potential diagnoses based on patient symptoms and medical history, LLMs assist clinicians

in considering many possibilities. This support helps in ruling out unlikely conditions and focusing on the most probable diagnoses. LLMs utilize natural language processing to interpret clinical notes and patient records, thereby ensuring that no critical information is overlooked during the diagnostic process[70].

LLMs can contribute to the development of personalized treatment plans by analyzing individual patient data, including genetic information, lifestyle, and comorbidities. These models can suggest tailored treatment options that are most likely to be effective for specific patients. Personalized medicine, supported by LLMs, ensures that treatment plans are not only based on general clinical guidelines but also customized to meet the unique needs of each patient[71]. This approach has been shown to enhance treatment efficacy and patient satisfaction[72].

2.4.2 Medical Research and Knowledge Discovery

Large Language Models (LLMs) have demonstrated exceptional capabilities in automating the process of literature review and summarization. These models can rapidly analyze extensive medical literature, identify the most relevant studies, and provide concise summaries of key findings[73]. This automation helps healthcare professionals stay up-to-date with the latest research without the need to manually sift through a vast amount of information. By leveraging LLMs, clinicians can make more informed decisions, improve patient care, and accelerate the translation of research into clinical practice[65].

LLMs provide healthcare professionals are continuously informed about the latest medical research. These models can be integrated into digital platforms or clinical decision support systems to provide real-time updates on new studies, clinical trials, and emerging treatments[74]. By delivering personalized research summaries and recommendations based on the clinician's area of specialization, LLMs help healthcare professionals stay ahead of the curve and provide their patients with the most up-to-date and evidence-based care[72].

another tasks LLMs can accelerate the drug discovery process by analyzing vast amounts of data, including chemical structures, genomic information, and biological mechanisms[75]. These models can identify promising drug candidates by recognizing patterns and relationships that may not be readily apparent to human researchers.

This capability helps to the drug discovery pipeline, reducing the time and cost associated with traditional methods[76].

Predicting Drug Interactions and Side Effects Predicting potential drug interactions and side effects is a crucial aspect of drug development and patient safety. LLMs can leverage their natural language processing abilities to analyze clinical trial data, patient records, and scientific literature to anticipate possible adverse reactions and drug-drug interactions[77]. By incorporating this information into the drug development process, LLMs can help researchers and clinicians make more informed decisions, leading to safer and more effective treatments[78].

2.5 Conclusion

The integration of Large Language Models (LLMs) in the medical field has the potential to greatly enhance various aspects of healthcare delivery. LLMs offer a wide range of applications, from improving clinical decision support systems to advancing telemedicine platforms, which can ultimately lead to better patient outcomes and more efficient healthcare processes.

Telemedicine platforms, powered by LLMs, enable remote consultations and real-time health monitoring, making healthcare more accessible, especially in underserved areas. Through the use of video conferencing tools, secure messaging platforms, and remote monitoring devices, continuous patient care and timely interventions can be achieved. Mobile health applications further complement these efforts by allowing for symptom tracking, medication reminders, and health behavior monitoring, thereby promoting greater patient engagement and adherence to treatment plans.

Furthermore, the application of AI algorithms for medical diagnosis highlights the transformative potential of LLMs. Machine learning models, such as Support Vector Machines, Random Forests, and Neural Networks, have demonstrated their effectiveness in diagnosing conditions and predicting patient outcomes. Deep learning models, particularly Convolutional Neural Networks, have shown remarkable success in medical image analysis, opening up new possibilities for early detection and precise treatment of diseases.

In summary, the adoption of LLMs in the medical field represents a significant ad-

vancement in healthcare technology. By leveraging these models, healthcare providers can improve diagnostic accuracy, streamline clinical workflows, and ultimately enhance patient care. As technology continues to evolve, the role of LLMs in medicine is expected to expand, paving the way for more innovative solutions and improved health outcomes.

Chapter 3

Theoretical Framework

3.1 Introduction

In the new era of healthcare, the accuracy and timeliness of diagnoses are crucial for effective treatment and management of medical conditions. However, the diagnostic process can be complex, especially in scenarios where symptoms present ambiguously or overlap. Traditional diagnostic methodologies heavily rely on the expertise of medical professionals, which is inherently bounded by individual knowledge and experience. These limitations can lead to misdiagnoses, delayed treatment, and suboptimal patient outcomes.

To tackle this challenge, we introduce an AI-powered medical assistant that harnesses the capabilities of large language models (LLMs) to facilitate the diagnosis process. LLMs, a class of artificial intelligence models, are trained on vast corpora of textual data, enabling them to comprehend and do human-like missions. This allows the medical assistant to engage in natural language conversations, understand user queries, and provide relevant diagnostic information based on the provided symptoms and contextual cues.

The integration of AI-powered technologies, such as LLMs, in healthcare has the potential to revolutionize the diagnostic process. By leveraging the vast knowledge and processing capabilities of these models, the medical assistant can complement and enhance the expertise of human medical professionals, leading to more accurate and timely diagnoses, and ultimately, improved patient care

3.2 System Requirements

To develop an effective medical AI assistant utilizing large language models (LLMs) for medical diagnosis, several system requirements need to be met:

- Chatbot Interface Development: The core of the system is a chatbot that simulates doctor-patient interactions. This involves creating an intuitive and user-friendly interface that can effectively interact with users, interpret their queries, and provide accurate disease predictions.

- Personalization Engine: To enhance the user experience, the system must incorporate a personalization sets. This component try to collects relevant personal details, such as medical history, age, gender, and other demographic information if possible, to customize the diagnostic responses to the individual user. This ensures that the advice and predictions are as relevant and specific as possible.

- LLM Configuration and Control: Proper configuration and control of the LLM are crucial to ensure the generation of relevant and context-appropriate responses. This includes setting appropriate parameters, provide the model based with medical data if availble, and implementing mechanisms to guide the model's outputs in line with medical standards and protocols.

- Input Preprocessing: To improve the accuracy and consistency of the diagnosis, user inputs must be preprocessed before being fed into the LLM. This involves techniques for reformulating and standardizing queries, such as correcting grammar, removing ambiguities, and ensuring clarity,breakdown query to enhance the model's understanding and response accuracy.

- Data Management: Efficient handling of the collected personal details and user inputs is essential. The system must integrate and manage this data well, ensuring it is readily accessible for personalized response generation while maintaining data integrity and performance.

By addressing these system requirements, the medical AI assistant can deliver a reliable, personalized, and accurate diagnostic experience for users.

3.3 Overvall architecture

The overview provides a conceptive description of the AI medical assistant system we developed, detailing its core functionality and the interactions between the user and the system. The system is designed as a conversational AI, leveraging large language models (LLMs) to assist users in diagnosing medical conditions through an intuitive and user-friendly interface.

The following use case diagram3.1 illustrates the interaction between the user and the system Enhancement module within the AI medical assistant system:

- ***the system Enhancement module*** : This module serves as the central component of our AI assistant. It is responsible for processing queries using various methods, which can be adjusted based on the approach employed.Also he is designed to be flexible, allowing for different query processing techniques to be implemented as needed.It also has the capability to configure and manage large language models (LLMs). This includes tasks such as adjusting model parameters affecting performance,andintegration with other methods.

- *Choose Method*: The user begins by selecting the preferred method of interaction. They can either enter a query directly or input their symptoms and personal details for another method approach.

- *Send Request*: The user then sends a request or query to the system. This could be a direct question about a medical condition or entered some additional details.

- *Generate Diagnosis*: Upon receiving the request, the system processes the information and generates an appropriate response. This response generation is facilitated by the LLM, which uses DATAs retreived from the knowledge base and provide a contextually accurate answer.

- *Reformulate Query*: the system enhancement module reformulate the user's query to better align with the available data or to clarify ambiguous information. This ensures that the LLM can provide the most accurate and relevant response.

- *Instruct LLM*: The system enhancement module provides specific instructions to the LLM, guiding it to generate a Diagnosis that adheres to the medical context

and user requirements.

The user initiates the *Choose Method of Diagnosis* and *Send Health Description status* use cases, starting the interaction with the system. The system agent is actively involved in the *Reformulate Query* and *Instruct LLM* use cases. The system enhancement module role is crucial in ensuring that user queries are correctly interpreted and that the LLM is properly guided to generate the response. The language model (LLM) performs the *Generate Diagnosis* use case, producing the final output based on the system enhancement module instructions.

The *Send Health Description status* use case extends to the *Reformulate Query* use case, indicating that the system may need to reformulate the user's query before it can be processed effectively by LLM. The *Reformulate Query* use case extends to the *Choose Method of Diagnosis*use case, suggesting that if the query reformulation indicates a different approach might be more suitable, the user may be prompted to choose a different method. The *Generate Response* use case includes the *Send Health Description status* use case, meaning that generating a response is contingent upon receiving and understanding the user's initial request.

This use case diagram3.1 encapsulates the core functionality of the AI medical assistant system, highlighting the interactions and processes that ensure users receive accurate and contextually relevant medical prediction . By structuring the system in this manner, we ensure a seamless and efficient user experience, leveraging advanced AI techniques to assist with medical diagnoses.

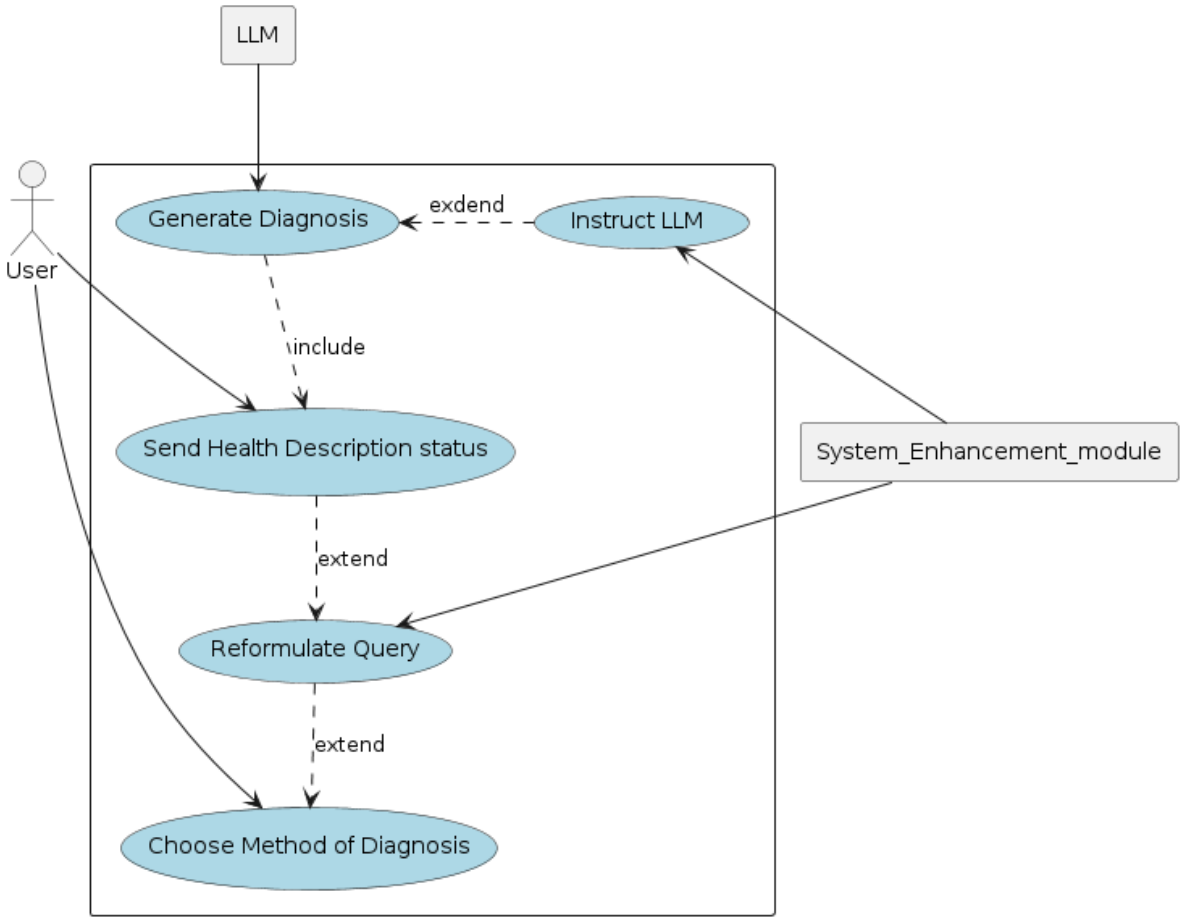


Figure 3.1: Use Case Diagram of system interactions.

To further dissect the idea, we present two derived use case diagrams to show the main differences between the two methods, as shown in Figure 3.2 and Figure3.3:

Enter Query and *Enter Required Details*. These use cases represent the two primary ways in which users can interact with the AI medical assistant system.

The *Enter Query* use case involves the user directly entering a query or question related to their medical condition. The system enhancement module processes this query, then passes it to the LLM to generate a diagnosis.

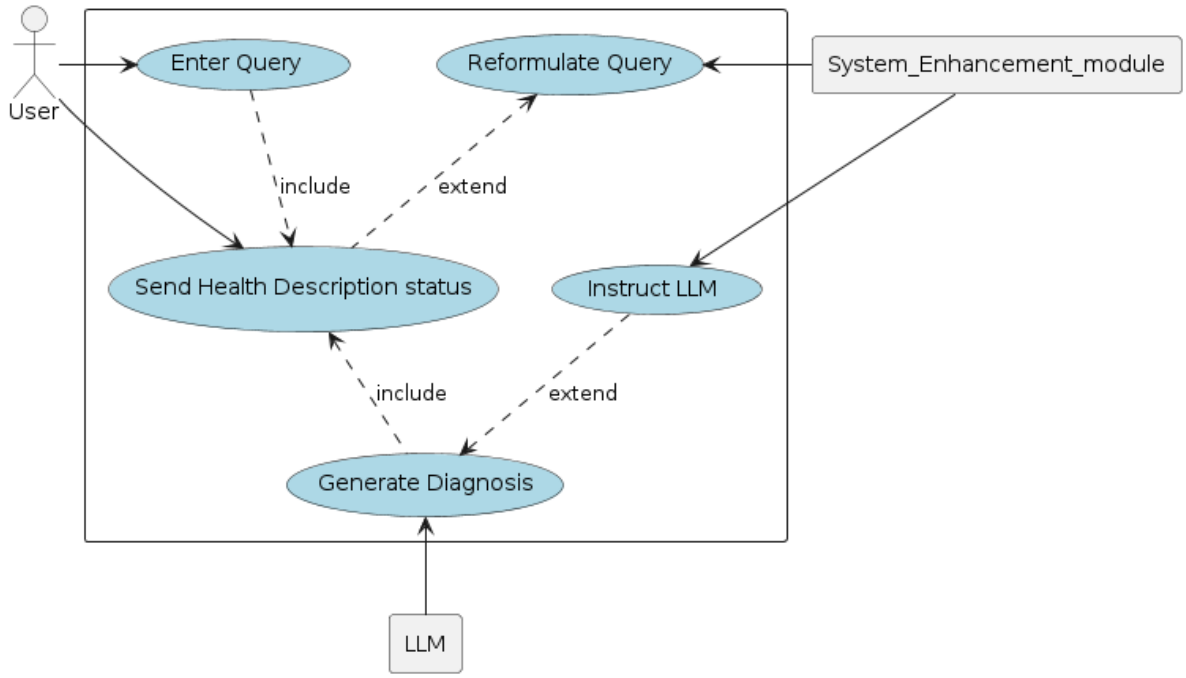


Figure 3.2: Native Diagnosis Use Case Diagram

The *Enter Required Details* use case ensures the user provides extra details about their medical condition, which are added in *Send Health Description*. This use case is designed to gather more detailed information by making the system enhancement module perform the *Process Required Details* use case, which can help the system generate a more accurate diagnosis.

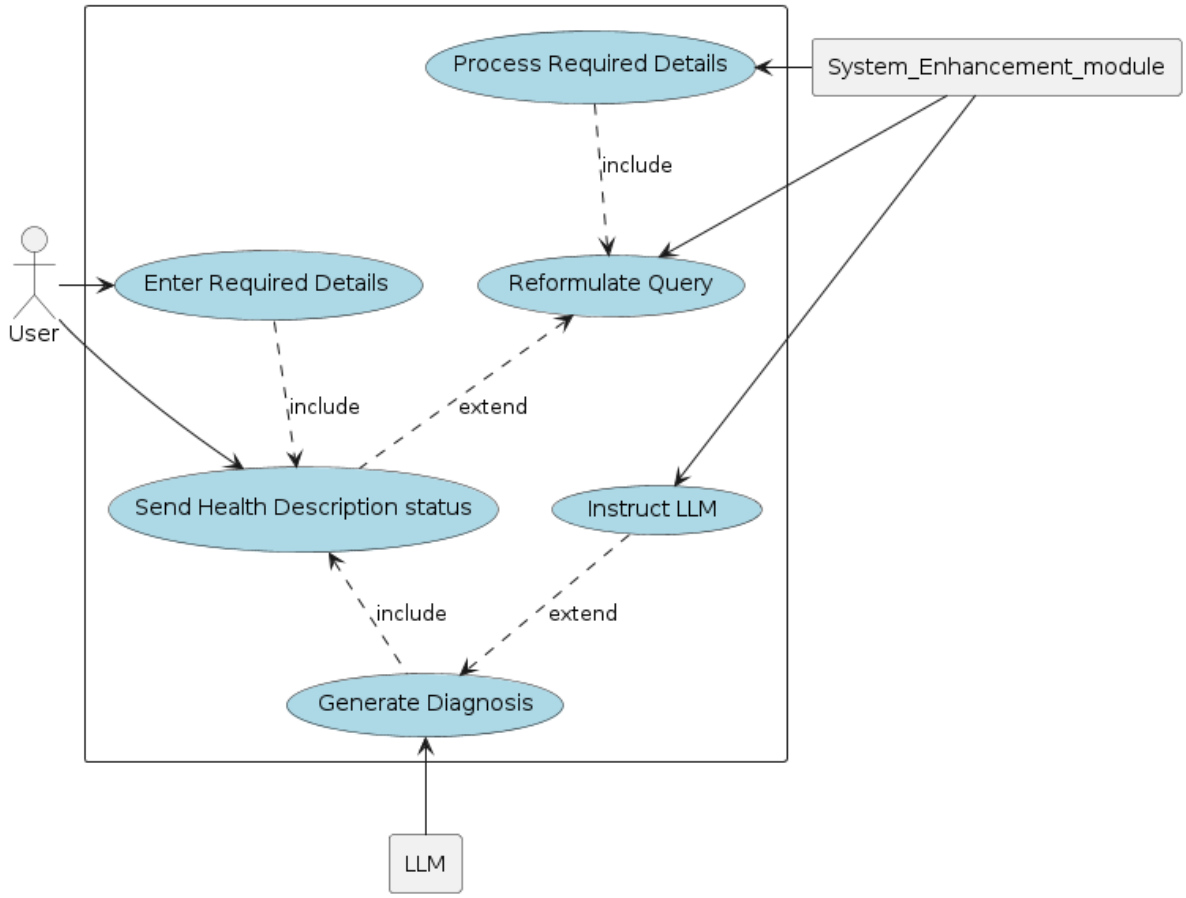


Figure 3.3: Assisted Diagnosis Use Case Diagram

3.4 Details descriptions

The basic idea behind this is to develop an AI-medical assistant that can predict potential diseases based on a user's query or description of symptoms. Instead of directly feeding the user's query to a Large Language Model (LLM), this approath very simple and efficiant.

We're going to rely the Retrieval-Augmented Generation (RAG) technique, which combines the power of retrieving relevant information from a knowledge base with the language generation capabilities of LLMs. Here's how it works:

- **User Query**: The user inputs their query or symptom description into the system.
- **Informations Retrieval:** Based on the user's query, we'll use an information retrieval system to fetch relevant medical knowledge as from a Vector database

or knowledge base. This could include information about diseases, symptoms, treatments, and other related medical data.

- **Prompt Construction**: Using the retrieved information, we'll construct a context or prompt that provides relevant background knowledge to the LLM. This context will contain the essential information needed to understand and accurately respond to the user's query.

- **LLM Generation**: We'll feed the constructed context or prompt into the LLM, which will impact its language understanding capabilities and the provided context to generate a response. In our case, the LLM will aim to predict potential diseases or conditions based on the user's symptoms and the relevant medical knowledge.

  By integrating the RAG in this approach, we're essentially giving the LLM a focused and relevant source of knowledge, tailored specifically to the user's query. This way, the LLM can better understand the context and provide more accurate and informed predictions, benefiting from the combination of retrieved medical knowledge and its own language generation capabilities.

Now, let's navigate through the intricacies of our two approaches:

3.4.1 Option 1: Native Diagnosis

The figure 3.4 illustrates our first approach, which relies solely on the language model's capability and user-provided context. This approach utilizes a Retrieval-Augmented Generation (RAG) technique by preprocessing a dataset containing medical information and setting it up as a vector database. The assistant initiates a conversational dialogue with the user, prompting them to enter a description. It diligently collects relevant symptoms and contextual nuances. Based on this input, the assistant generates a list of potential diseases or conditions, accompanied by succinct descriptions and alignment indicators showcasing symptom congruence. To break down the workflow of the system, we present a map explaining it:

- **1.** User Interaction: The user inputs a query describing symptoms or conditions.

- **2.** Query Processing: The system sends the user's query to an embedding model.

- **3.** Embedding: The embedding model converts the query into an embedded format.
- **4.** Vector Database Retrieval: The embedded query is used to retrieve relevant context from the vector database.
- **5.** Contextual Enrichment: The retrieved context is combined with the original query to provide comprehensive information.
- **6.** Response Generation: A response is generated and sent back to the user, offering potential diagnoses and relevant information.

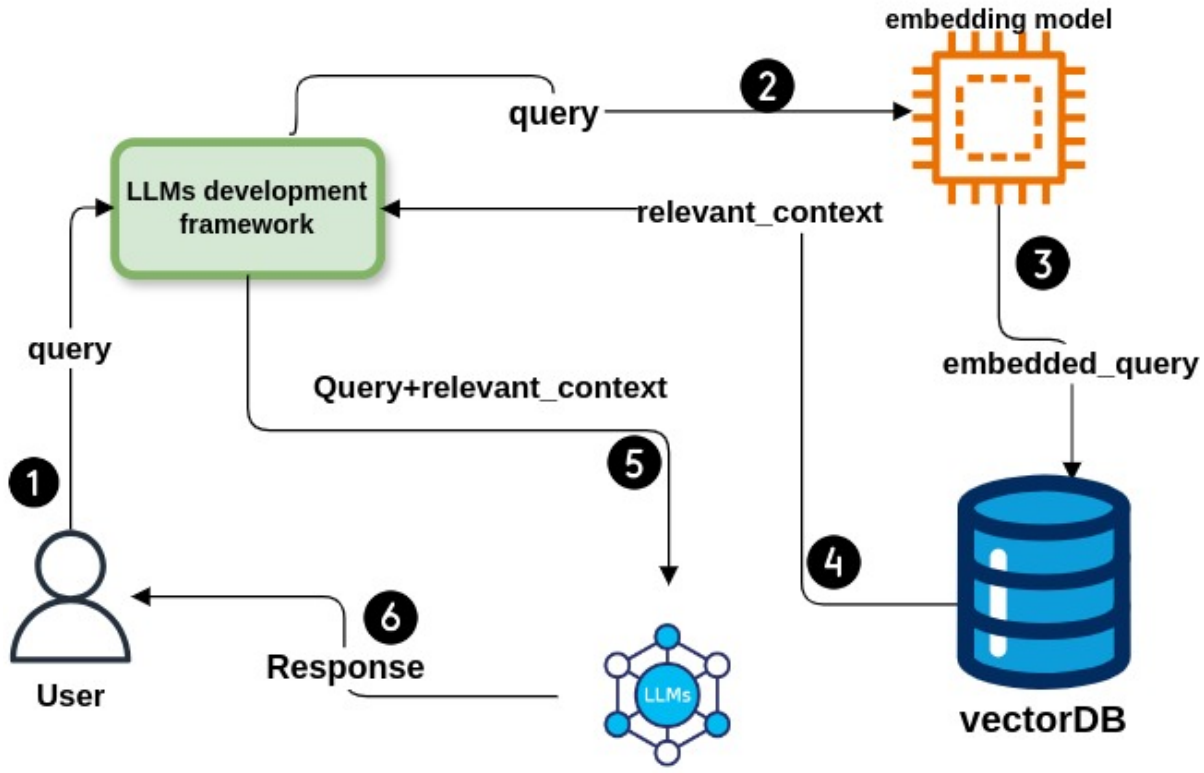


Figure 3.4: Workflow Diagram of System Interactions case.N°1.

The assistant taps into the stored Vector Database to enhance its comprehension and provide more precise diagnoses, epitomizing a synergy between cutting-edge AI and robust medical knowledge bases.

3.4.2 Option 2: Assisted Diagnosis

Contrary to the independent nature of Option 1, our second approach fuses external APIs, which can also be considered a RAG approach to enhance our system. Particularly, a symptom checker API is used to augment the diagnostic process. The assistant, following a conversational paradigm, prompts the user to provide symptoms, gender, and year of birth. Subsequently, leveraging the symptom checker API, the assistant retrieves a collection of potential diagnoses adjusted to the user's input.

Moreover, the assistant harnesses the API's wealth of detailed health information to enrich the diagnostic discourse, fostering a holistic understanding of the user's health profile. Much like Option 1, the dataset complements the API-integrated diagnoses, providing an augmented knowledge base to refine diagnostic recommendations. Here is figure 3.5 and the map explaining:

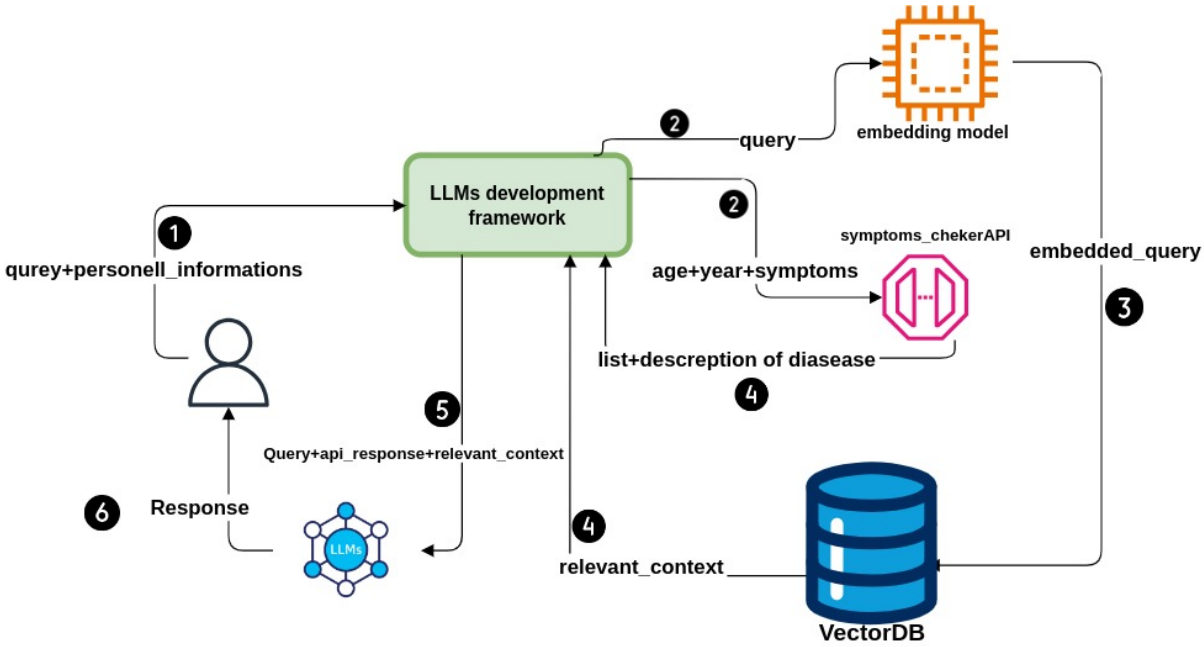


Figure 3.5: Workflow Diagram of System Interactions case.N°2.

- **1.** User Interaction: The user inputs a query that describes their condition along with personal information such as gender and year of birth.
- **2.** Query Processing and API Call: The system processes the query and sends it to the embedding model, and the personal information to a symptom checker API.

- **3.** Embedding: The embedding model converts the query into an embedded format for semantic search.
- **4.** Vector Database Retrieval and API Response: The API returns a list of potential diagnoses and descriptions based on the user's input, and relevant context is retrieved from the vector database.
- **5.** Contextual and API-Response Enrichment: The original query, API response, and relevant context are combined and given to the LLM model.
- **6.** Response Generation: A comprehensive response, breaking down the user's medical conditions with logical explanations, is generated by the LLM and sent back to the user.

3.4.3 Datasets

a. WebMD Dataset

The WebMD Dataset consists of medical data sourced from WebMD, a popular online resource for medical information. This dataset includes various types of medical information such as symptom descriptions, conditions, medications,paitient-doctor conversation and user reviews.it can be usead as medical knowledge The data is valuable for its comprehensive coverage of common medical conditions and treatments [79].this data source are the main component to act like Vector Database for our architectures.

3.5 Evaluation

Evaluating the performance of our chatbot is a critical step to ensure its reliability and effectiveness in providing accurate medical diagnoses. The evaluation process involves comparing the responses generated by the LLM model to real-world diagnoses provided by medical professionals. This approach aims to measure how well the chatbot can deliver correct diagnoses based on user-reported symptoms.

The primary goal of this evaluation is to determine the accuracy of our chatbot in diagnosing medical conditions compared to actual medical outcomes. This involves

analyzing and testing the chatbot's responses to various user inputs and comparing them to established medical diagnoses to see if the chatbot provides correct and relevant information.
In addition to evaluating our custom chatbot, we also applied the same evaluation process to GPT-3.5 to benchmark its performance against our chatbot. By calculating the accuracy of diagnoses provided by GPT-3.5 and comparing it to our chatbot, we aim to verify the improvements and effectiveness of our approach. This comparative analysis helps in identifying the strengths and weaknesses of both models, providing a comprehensive understanding of their diagnostic capabilities in real-world scenarios.

The key metric used to assess the chatbot's performance is accuracy. Accuracy is defined as the proportion of correct diagnoses made by the chatbot out of the total number of cases evaluated. This metric provides a straightforward and quantifiable measure of the chatbot's diagnostic capabilities.

In our evaluation, we performed two methods: the first method involved using a custom criteria evaluator, and the other method involved calculating the semantic similarity using the model 'dmis-lab/biobert-base-cased-v1.1' on both the chatbot and GPT-3.5 models.

3.5.1 Semantic Similarity

Semantic similarity measures how similar two pieces of text are in terms of meaning. In the context of medical diagnosis, this involves comparing the generated responses from the chatbot to the reference diagnoses to see how closely they match in terms of meaning, rather than exact wording.
For this purpose, we used the model *'dmis-lab/biobert-base-cased-v1.1'*, which is specifically designed for biomedical text. This model helps in quantifying the semantic similarity between the chatbot's responses and the reference diagnoses. The higher the semantic similarity score, the closer the chatbot's response is to the actual medical diagnosis in terms of meaning.

The procedure involves encoding both the chatbot's response and the reference diagnosis into vectors using the BioBERT model, and then calculating the cosine similarity between these vectors. This similarity score is then used to assess the accuracy

of the chatbot's diagnosis in a more nuanced manner, capturing the semantic essence of the responses.

3.5.2 LangChain String Evaluators

LangChain's string evaluators are flexible tools designed to assess the quality of textual outputs generated by AI models. There are several types of string evaluators, such as Exact-Match,embedding distance and question answering evaluators, each tailored to different use cases, including chatbots [80].

The Exact-Match evaluator measures the degree to which the generated response matches a predefined correct answer exactly. While this approach is useful for tasks requiring precise answers, it may not be suitable for evaluating medical diagnoses, where multiple correct diagnoses might exist for a given set of symptoms.

The question answering evaluator assesses an AI model's ability to provide accurate and relevant answers to specific questions,this commonly used in testing RAG models to evaluate the quality of the generated output by yhe AI or LLMs. However, this type of evaluator might fall short in scenarios where the complexity and variability of medical diagnoses require nuanced understanding and interpretation.

The criteria evaluator is used for a more nuanced evaluation by considering various aspects of the generated responses, such as *Conciseness*, *Coherence*, *Harmfulness*, *Maliciousness*, and *Misogyny*. However, this approach is not suitable for our methodology of evaluation because these criteria assess the output from a general perspective, rather than focusing specifically on whether the generated output is medically accurate or relevant.

Due to these limitations, we employ a custom criteria evaluator to better suit our needs for medical diagnosis evaluation.

Custom Criteria Evaluator

The custom criteria evaluator allows for a more focused and detailed assessment of the chatbot's diagnostic capabilities. This evaluator uses user-defined benchmarks that address specific aspects of the generated text relevant to the use case. In our context, these criteria include the accuracy of medical diagnoses, the appropriateness of medical advice, and the clarity of the diagnosis provided.

For our evaluation, we use a dataset containing queries with reference diagnoses. The process involves providing the LLM model with these queries, generating responses, and then comparing these responses to the reference diagnoses. The custom criteria evaluation assigns values of Y (Yes) or N (No) and scores of 0 or 1 based on whether the LLM model correctly predicts the diagnosis[80].
Here is an overview of the evaluation procedure as illustrated in Figure 3.6:

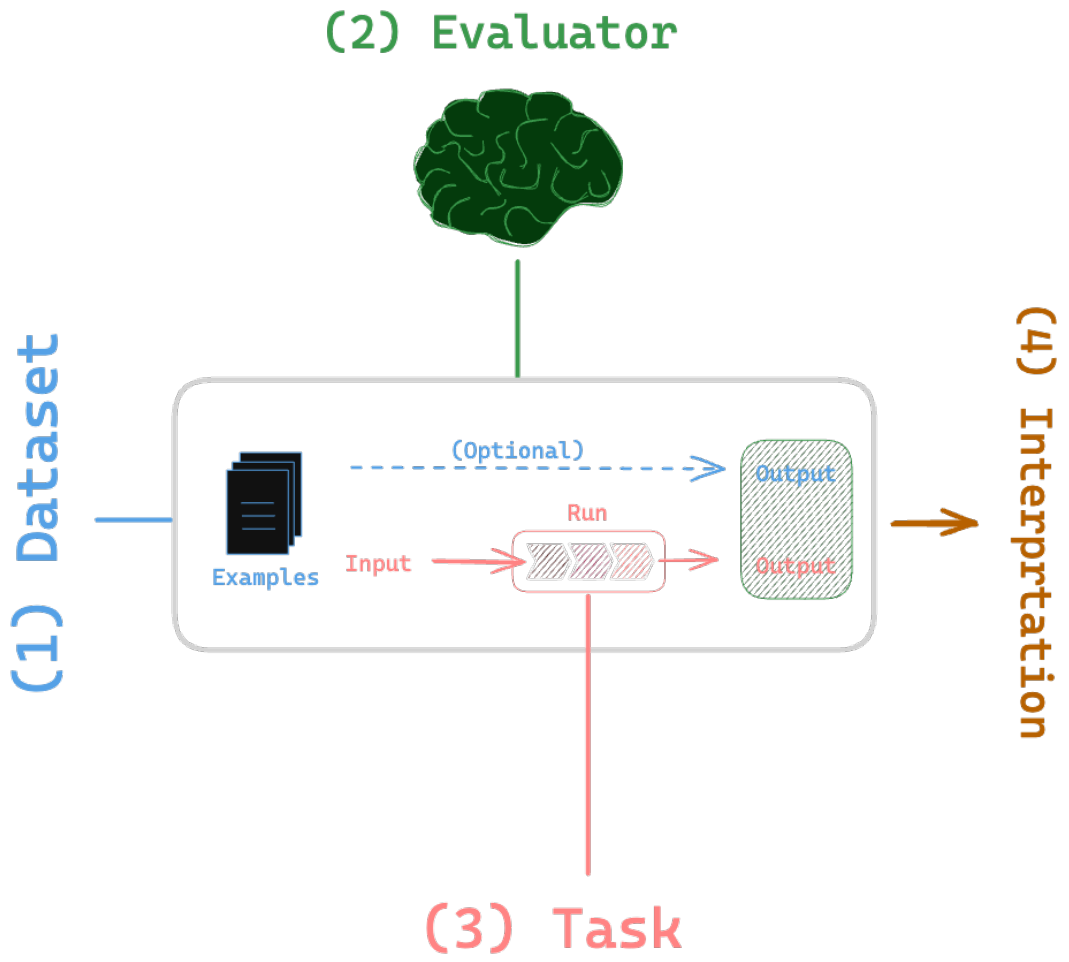


Figure 3.6: Evaluation Overview
[80]

The custom criteria evaluator systematically compares the AI-generated responses to actual medical diagnoses, ensuring that the responses are not only accurate but also clear, relevant, and useful to the user. This comprehensive evaluation helps maintain a high standard of quality in the chatbot's interactions.

3.6 Conclusion

In this chapter, we have established a comprehensive theoretical look for the development and evaluation of an AI-powered medical assistant aimed at enhancing the diagnostic process in healthcare. The introduction underscored the critical need for accurate and timely diagnoses and the inherent limitations of traditional methodologies. By leveraging large language models (LLMs), we propose an innovative solution to complement and augment the expertise of medical professionals, thereby improving diagnostic accuracy and patient outcomes.

We detailed the essential system requirements necessary for implementing this AI medical assistant, emphasizing the importance of a user-friendly chatbot interface, a robust personalization engine, effective LLM configuration and control, thorough input preprocessing, and efficient data management. These components are crucial to creating a reliable and effective diagnostic tool.

The overall architecture of the system was described, highlighting the interactions between users and the AI medical assistant through a conversational interface. The use case diagrams illustrated the core functionalities and the flow of information within the system, demonstrating how user queries are processed and translated into accurate diagnostic responses.

Additionally, we introduced two primary approaches for diagnosis: the Native Diagnosis approach, which relies solely on the LLM's capabilities, and the Assisted Diagnosis approach, which integrates external symptom checker APIs to enhance the diagnostic process. By comparing these approaches, we aim to identify the most effective method for integrating AI into medical diagnostics.

Through the detailed descriptions of these approaches, particularly the application of the Retrieval-Augmented Generation (RAG) technique, we have outlined how the AI medical assistant can harness both the power of LLMs and external medical knowledge to provide informed and contextually relevant diagnoses.

This theoretical framework sets the stage for the practical implementation and evaluation of the proposed AI medical assistant. By addressing the identified system requirements and carefully designing the architecture, we can develop a tool that not only assists in medical diagnosis but also represents a significant step forward in the

application of AI in healthcare.

Chapter 4

Operational framework

4.1 Introduction

This chapter presents a detailed overview of the operational framework developed for an advanced medical chatbot. It begins by introducing the various tools and environments used in the development process, such as Google Colab and Python libraries, which provide the foundational infrastructure for coding and computation. The chapter then delves into the specific frameworks and toolkits that power the chatbot's capabilities, including LangChain, FastAPI, and Pinecone, highlighting their roles in managing prompts, handling data, and facilitating model interactions. Additionally, the integration of external APIs, such as ApiMedic, is discussed to showcase how the chatbot can access and utilize extensive medical knowledge bases. Finally, the chapter outlines the implementation details, from data preprocessing to user interaction, and concludes with an evaluation of the chatbot's performance, emphasizing the accuracy and reliability of its diagnostic outputs.

4.2 Development Environment

Google Colab

Google Colaboratory, commonly known as Google Colab, is a free, cloud-based platform that allows users to write and execute Python code in a web-based, Jupyter notebook environment. It provides free access to computing resources, including GPUs,

which makes it particularly useful for machine learning and data analysis tasks [81].

4.2.1 Python

Python is a high-level, interpreted programming language known for its simplicity and readability. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is widely used in various domains such as web development, data science, artificial intelligence, and scientific computing due to its extensive standard library and active community [82].One of the main libraries that we used to reach our work is:

I. Transformers

The Transformers library, developed by Hugging Face, is a powerful library for natural language processing (NLP). It provides a unified interface for using pre-trained models for various NLP tasks, such as text classification, question answering, and language translation. The library supports a wide range of transformer architectures, including BERT, GPT, and T5 [83].

II. Datasets

The Datasets library, also developed by Hugging Face, is a lightweight library for accessing and sharing datasets. It is designed to be efficient and easy to use, providing tools for loading, preprocessing, and manipulating large datasets for machine learning and data science projects [84].

III. OpenAI

The OpenAI library provides a Python interface to OpenAI's powerful language models, such as GPT-3 and GPT-4. It allows developers to integrate these models into their applications for tasks like text generation, summarization, and translation. The library simplifies the process of making API calls to OpenAI's services [85].

ApiMedic API

The ApiMedic API provides a symptom checker and medical diagnosis service. It is designed to be integrated into applications to provide users with information about potential **medical conditions** based on their **symptoms**. The API leverages a comprehensive medical knowledge base and advanced algorithms to deliver accurate and

reliable diagnostic informations [86].

Flutter

Flutter is an open-source UI software development toolkit created by Google. It is used to develop cross-platform applications for mobile, web, and desktop from a single codebase. Flutter's framework includes a rich set of pre-designed widgets and tools that enable developers to build visually attractive and performant applications [87].

Frameworks and Toolkits

I. LangChain

LangChain is an open-source framework designed to facilitate the development of applications powered by large language models (LLMs). Its primary goal is to simplify the complex processes involved in integrating and utilizing language models, providing a user-friendly interface and a robust set of tools.

LangChain offers advantanges features:

- **Prompt Management**: LangChain provides tools for managing prompts effectively. It allows developers to create, store, and modify prompts easily, ensuring that they can leverage the full potential of LLMs in various applications.
- **Data Handling**: The framework includes capabilities for seamless data integration and preprocessing. It supports different data formats and sources, enabling developers to feed diverse datasets into their language models efficiently.
- **Model Interaction**: LangChain simplifies the interaction with multiple language model providers. It abstracts the complexities of model API interactions, making it straightforward to switch between or combine different models as needed.
- **Customizability**: The framework is highly customizable, allowing developers to tailor its components to fit their specific use cases. This flexibility is particularly beneficial for building specialized applications in areas like natural language understanding, text generation, and conversational AI.

- **Integration**: LangChain can be integrated with other popular frameworks and tools in the machine learning ecosystem, such as TensorFlow, PyTorch, and Hugging Face Transformers. This interoperability enhances its utility in diverse development environments.

LangChain has gained traction in the developer community due to its ability to streamline the development process for language model-based applications. By providing a comprehensive set of utilities and abstractions, it enables developers to focus on creating innovative solutions without getting bogged down by the intricacies of model management and data processing.[88]

> "LangChain is designed to make the development of language model applications easier, more productive, and more reliable." — *LangChain Documentation*

.



Figure 4.1: Langchain Modules
[88]

II. FastAPI

FastAPI is a modern, fast (high-performance), web framework for building APIs with Python 3.7+ based on standard Python type hints. It is designed to be easy to use and

to provide high performance. FastAPI allows developers to quickly create robust and efficient web APIs, with automatic generation of interactive API documentation [89].

III. Uvicorn

Uvicorn is a lightning-fast ASGI server implementation, using `uvloop` and `httptools`. It is designed to serve fast HTTP requests and support the development of asynchronous web applications. Uvicorn is often used in conjunction with FastAPI to provide a high-performance, production-ready web server [90].

IV. Pinecone

Pinecone is a scalable hybrid cloud storage system that combines the benefits of cloud storage and local storage to provide efficient and reliable data storage. It is designed to address the challenges of large-scale data management, including data growth, heterogeneous storage environments, and the need for high performance and availability[91].

4.3 Implementation Details

4.3.1 Chatbot Implementation

The core component of our system is a conversational AI chatbot powered by OpenAI's ChatGPT model. The implementation of the chatbot involves the following steps:

a. Data Preprocessing

We begin by loading a dataset containing medical questions and answers from Huggingface platform. The preprocess text function is applied to clean and preprocess the text data, removing specific substrings and replacing them with appropriate delimiters.

b. Sentence Embedding

To convert the text data into numerical representations suitable for vector similarity operations, we employ the Sentence Transformer model. Specifically, we utilize

the ”sentence-transformers/all-MiniLM-L6-v2” model, which generates dense vector embeddings for the DATAs in the dataset.

c. Vector Database Setup

After generating the embeddings, we set up a vector database using Pinecone, a vector database service. We establish a connection to the Pinecone API and create an index named ”medical source Knowlegde” to store our data. The dataset is then batched and uploaded to the Pinecone index, associating each row with its corresponding embedding vector and metadata.

d. Vector Search and Retrieval

To test the vector database retrieval, we define a sample query string, encode it using the Sentence Transformer model, and perform a vector similarity search on the Pinecone index. The top-k most similar results are retrieved, along with their corresponding scores and contexts.

e. Conversational Integration

We integrate the OpenAI ChatGPT model to assist with diagnosis based on the retrieved medical information from the vector database. After setting up the OpenAI API key and creating a ChatOpenAI instance with desired parameters, we initialize a conversation with the ChatGPT model, providing it with instructions to act as an AI-assisted doctor and diagnose the patient’s symptoms based on the provided information.

f. Query Processing and Response Generation

For each user query, we retrieve relevant medical information from the vector database by encoding the query and performing a vector similarity search. We then construct an augmented prompt that includes the retrieved medical information and the user’s query. This augmented prompt is sent to the ChatGPT model, which generates a diagnosis response based on the provided information and its own knowledge.

The integration of the vector database and the conversational AI model allows our chatbot to leverage both external medical knowledge sources and its own language understanding capabilities to provide accurate and informative diagnoses tailored to the user's input.

4.3.2 Symptom Checker API Integration

To enhance the capabilities of our chatbot and provide more accurate and informative medical diagnoses, we integrated a third-componnent symptom checker API. This API offers a range of functionalities, including retrieving a list of available symptoms, obtaining potential diagnoses based on the provided symptoms and user information, and retrieving detailed information about specific health issues.
The integration process involved several steps:

a. API Setup and Authentication

We first established a connection with the symptom checker API by configuring the necessary API endpoints and credentials. This included setting up the authentication process, which involved generating a secure hash using the provided API key and secret key, and obtaining an authentication token.

b. Retrieving Symptoms and Diagnosis

With the authenticated connection, we implemented functions to retrieve a list of available symptoms from the API. Additionally, we developed a function to obtain potential diagnoses based on the user's input, which included their specified symptoms (represented as IDs), gender, and year of birth.

c. Retrieving Issue Information

To provide more detailed information about the predicted diagnoses, we implemented a function that retrieves comprehensive details about specific health issues by sending requests to the API's issue information endpoint with the corresponding issue ID.

d. User Interaction and Diagnosis Output

In the main program flow, we first obtained an authentication token from the API. Then, we prompted the user to enter their gender, year of birth, and a comma-separated list of symptoms they are experiencing. We mapped the user-entered symptom names to their respective IDs using the list of available symptoms retrieved from the API.

e. Diagnosis Prediction and Output

With the user's symptoms, gender, and year of birth, we called the diagnosis function to obtain a list of potential diagnoses from the API. Finally, we presented the predicted diagnoses to the user, along with their descriptions, by iterating over the diagnosis list and retrieving the issue information for each predicted diagnosis.

By integrating the symptom checker API, our chatbot can leverage the power of a specialized medical knowledge base to provide more accurate and informative diagnoses based on the user's input. This integration enhances the overall functionality and reliability of our chatbot, making it a more valuable resource for users seeking medical advice or information.

By combining the implementation details of the chatbot and the symptom checker API integration, we have developed a helpful system that can assist users in diagnosing potential health issues based on their reported symptoms and provide relevant medical information to support the diagnosis.

4.4 Graphical User interface

to make a gui for the user to help him use in easy way our chatbot,First we truned our chatbot to a Fastapi app who splitted into endpoints URIs who expect a request JSON from user and he reply also with JSON , after that we develloped interfaces using Flutter-SDK,then we connected the Flutter Dart code to the Fastapi server, here the breakingdown of the app interfaces discussing the possible interactions of the User :

I. Login interface



Figure 4.2: Login Interface

The login interface is the first screen the user encounters. It features a simple design, creating a welcoming atmosphere. The user is expected to enter their username and password in dedicated input fields. A blue "Login" button is prominently displayed for user authentication.The authentication interface of our web application serves as the initial point of interaction for users to access the system.

II. Method of diagnosis selection interface

This interface presents the user with two options for diagnosis: Assisted Diagnosis and Native Diagnosis. The interface is visually appealing.The user can choose their preferred method by tapping on the corresponding button.



Figure 4.3: Method Selection Interface

III. Assisted Diagnosis interface

This interface guides the user through a step-by-step diagnosis process. It starts by asking the user to provide basic information like gender, Yearofbirth and symptoms as we see in the Figure4.4. The user can search for symptoms using a search bar and select from a list of options(figure4.5). The interface also force the user to enter their year of birth and describe their condition in detail. A blue ”Submit” button allows the user to proceed with the diagnosis.

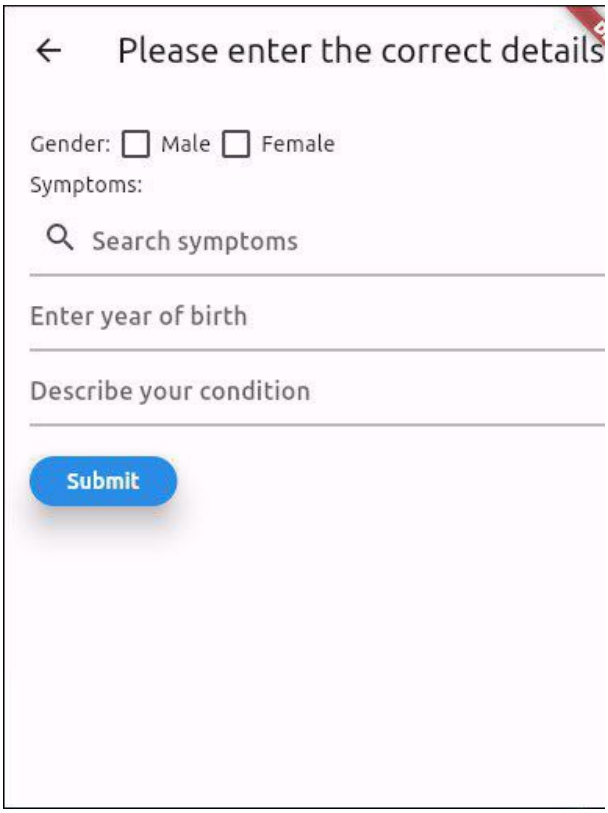


Figure 4.4: Assisted Diagnosis Interface

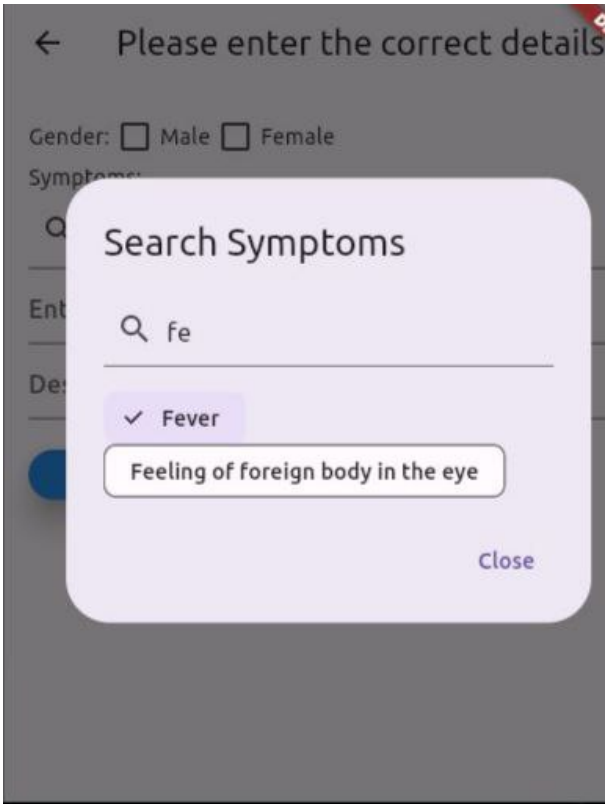


Figure 4.5: Selecting symptoms Tags

IV. Native Diagnosis interface

This interface allows the user to directly input their symptoms and receive a diagnosis. It features a simple design with a doctor and medical tools in the background. The user is prompted to enter their description in a text field. Buttons for Cancel and Send are provided for user interaction.

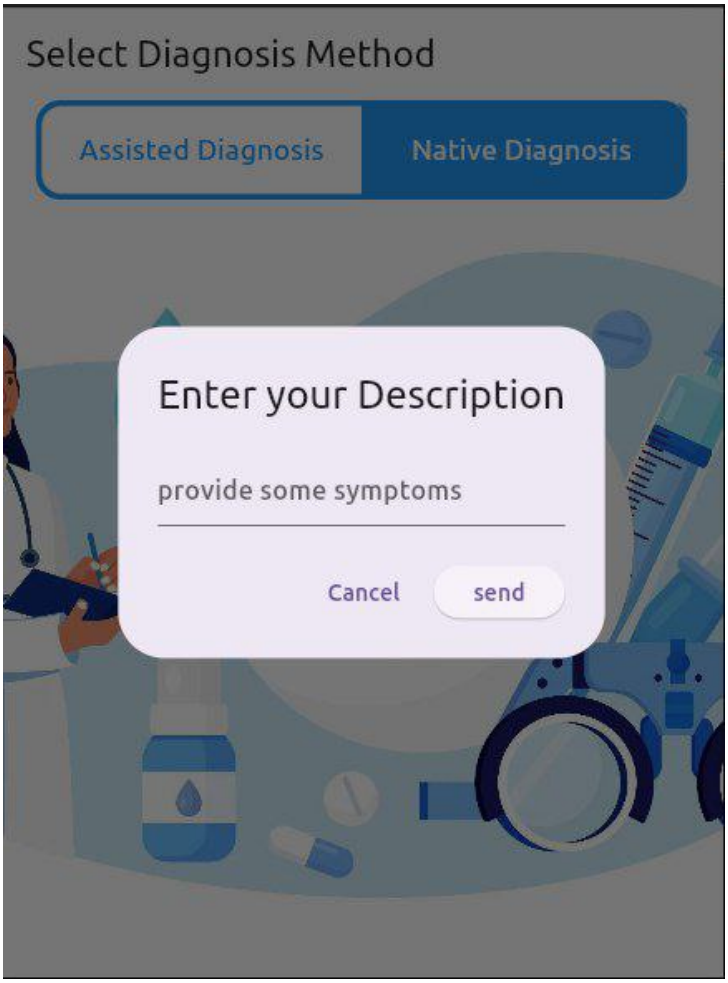


Figure 4.6: Native Diagnosis Interface

V. Conversation interface

This interface shown in Figure4.7 facilitates a conversational interaction between the user and the chatbot. The user can input their query in a text field and send it using a ”Send” button. The chatbot’s response is displayed in a separate area, providing information and guidance. The interface features a visual representation of the chatbot, enhancing the user experience. The user can ask follow-up questions or provide more

information to refine the conversation.

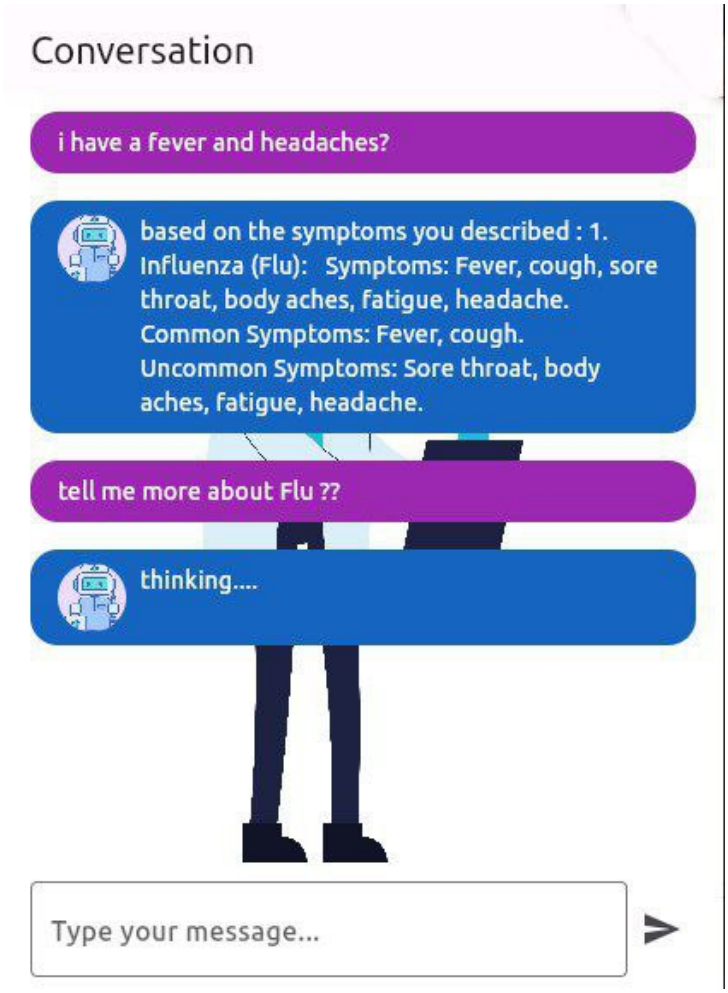


Figure 4.7: Conversation Interface

4.5 Evaluation

In the evaluation process and implementation, we apply the evaluation exclusively to the native diagnosis. The assisted diagnosis method wasn't applicable to the evaluation process and methods :

4.5.1 Results and Discussions

In this section, we present the results of our evaluation methods to assess the diagnostic accuracy of our chatbot compared to GPT-3.5-turbo. The evaluation comprises two methods: custom criteria evaluation and semantic similarity analysis. Both methods

applied the same dataset of 120 queries to ensure a consistent and fair comparison. The outcomes of these evaluations are discussed in detail below:

a) First Method

The first evaluation method involved calculating the semantic similarity between the responses generated by the chatbot and GPT-3.5-turbo against the reference diagnoses. Semantic similarity was computed using the 'dmis-lab/biobert-base-cased-v1.1' model, which is tailored for biomedical text.

For this evaluation, we considered a cosine similarity score threshold of 0.6. Scores above this threshold were deemed as similar, indicating a correct diagnosis. The accuracy was then computed based on the proportion of responses that met or exceeded this similarity threshold.

The accuracy scores obtained from the semantic similarity evaluation are presented in Table 4.1:

Model	Accuracy
Chatbot	73%
GPT-3.5-turbo	42%

Table 4.1: Accuracy scores based on Semantic Similarity for Chatbot and GPT-3.5-turbo

The results indicate that our chatbot achieved a semantic similarity-based accuracy of 73%, while GPT-3.5-turbo scored 42%. This demonstrates that our chatbot not only performs better in custom criteria evaluations but also in terms of semantic similarity to reference diagnoses. The significant improvement in accuracy highlights the effectiveness of our approach in providing accurate medical diagnoses, capturing the semantic essence of medical responses more effectively than GPT-3.5-turbo.

b) Second Method

In our evaluation, we performed custom criteria evaluation using 120 queries on both our chatbot and GPT-3.5. For this evaluation, we employed Anthropic Claude 3 Haiku as the judge, as it is known for its superior reasoning capabilities among free models. Each response generated by the LLM models was compared to the reference diagnoses,

and the custom criteria evaluation assigned values of Y (Yes) or N (No) and scores of 0 or 1 based on the accuracy of the predictions. The use of Claude 3 Haiku model as the judge ensures a high standard of reasoning and fairness in the evaluation process. Additionally, we set the parameter *temperature = 0* to sharpen the reasoning task.This approach try to cover the manual phase in *Method 1* to do auto-evaluations processes. The accuracy scores obtained from the evaluation are presented in Table 4.2:

Model	**Accuracy**
Chatbot	73.4375%
GPT-3.5-turbo	41.8803%

Table 4.2: Accuracy scores of Chatbot and GPT-3.5-turbo

The results clearly demonstrate that our custom chatbot significantly outperforms GPT-3.5-turbo in diagnostic accuracy. Our chatbot achieved an impressive accuracy rate of 73.44%, compared to GPT-3.5-turbo's accuracy of 41.88%. This substantial difference of 31.56% highlights the effectiveness of our approach in delivering accurate medical diagnoses.

This superior performance can be attributed to the precise configuration of the large language model (LLM) parameters. By fine-tuning settings such as *temperature*, *top-p*, and *frequency-penalty*and more, we optimized the model for integration with Retrieval-Augmented Generation (RAG) techniques. This careful adjustment of parameters ensured that our chatbot could provide more reliable and accurate diagnostic results.

To further understand the diagnostic accuracy, we analyzed the number of correct diagnoses identified by both methods. The outcomes are summarized as follows:

- Number of correct diagnoses by both the chatbot and GPT-3.5: 38
- Number of correct diagnoses by the chatbot but not by GPT-3.5: 49
- Number of correct diagnoses by GPT-3.5 but not by the chatbot: 11

While our chatbot significantly outperforms GPT-3.5-turbo overall, providing a higher number of unique correct diagnoses,there were 31 false prediction in diagnosis more specifically 11 cases where GPT-3.5-turbo made correct diagnoses that the

chatbot did not. These discrepancies can be attributed to certain limitations in our chatbot's model, such as potential gaps in its retrieved data, less optimal handling of ambiguous queries, or differences in how model interprets and prioritizes medical information. Understanding these limitations provides valuable insights for future improvements to enhance the chatbot's diagnostic accuracy.

4.6 Conclusion

In summary, this chapter has outlined the development and implementation of a comprehensive operational framework for an advanced medical chatbot. Leveraging a combination of cutting-edge technologies, including large language models, cloud-based environments, and specialized APIs, the framework ensures high accuracy and user-friendly interaction. The integration of detailed evaluation methods and custom criteria has demonstrated the chatbot's superior performance in diagnostic accuracy compared to other models. This robust setup highlights the potential for future improvements and broader applications in medical diagnostics and other domains.

General Conclusion

The thesis presents a comprehensive study on the design and development of a medical assistant application that leverages large language models (LLMs) to revolutionize healthcare services. The primary objective is to create an intelligent system that assists healthcare professionals and patients by facilitating large language models conversations, answering queries, and providing personalized healthcare recommendations.

The research explore into the capabilities of cutting-edge language models, showcasing their potential to transform healthcare delivery significantly. It emphasizes three critical areas: the system architecture, the data processing techniques employed, and the integration of domain-specific medical knowledge. By meticulously addressing these aspects, the thesis provides a robust framework for developing a medical assistant that is both efficient and reliable.

A significant portion of the work is dedicated to the evaluation and testing of the medical assistant. Rigorous assessments, including tests on real-world clinical scenarios, are conducted to demonstrate the system's effectiveness in enhancing healthcare outcomes. These evaluations are crucial as they validate the assistant's capabilities, ensuring that it meets the high standards required in the medical field. The results underscore the assistant's potential to positively impact healthcare by improving diagnostic accuracy, facilitating better patient management, and providing timely and relevant medical information.

The thesis also explores the challenges associated with deploying LLMs in healthcare applications. These include issues related to data quality and availability, accuracy and reliability of the models, and the need for transparency and interpretability in their decision-making processes. Addressing these challenges is essential for the successful integration of LLMs into healthcare, ensuring that the technology is both safe and effective for clinical use.

In conclusion, this research highlights the transformative potential of large language models in the healthcare domain. By leveraging it advanced natural language processing capabilities, the medical assistant developed in this study offers a promising solution to several pressing challenges in healthcare. It aims to enhance healthcare delivery, improve patient care, and provide accessible, personalized healthcare information. The work underscores the significance of integrating artificial intelligence into healthcare systems, paving the way for more innovative and efficient healthcare solutions in the future.

Transformers, introduced by *Vaswani et al.*[4], have emerged as a prominent architecture for LLMs due to their parallelization capabilities and effectiveness in capturing long-range dependencies in text. The self-attention mechanism employed in transformers allows the model to weigh the importance of different words in a sequence, facilitating better contextual understanding[9].

References

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[3] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[5] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[7] Ye Zhang and Byron Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*, 2015.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine trans-

lation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[9] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[10] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[11] Siyu Heng and Pamela A Shaw. A model-free and finite-population-exact framework for randomized experiments subject to outcome misclassification via integer programming. *arXiv preprint arXiv:2201.03111*, 2022.

[12] Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. Dialogpt: Large-scale generative pre-training for conversational response generation. *arXiv preprint arXiv:1911.00536*, 2019.

[13] Yi-Chieh Lee, Naomi Yamashita, and Yun Huang. Designing a chatbot as a mediator for promoting deep self-disclosure to a real mental health professional. *Proceedings of the ACM on Human-Computer Interaction*, 4(CSCW1):1–27, 2020.

[14] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. Deep knowledge tracing. *Advances in neural information processing systems*, 28, 2015.

[15] Dadi Ramesh and Suresh Kumar Sanampudi. An automated essay scoring systems: a systematic literature review. *Artificial Intelligence Review*, 55(3):2495–2527, 2022.

[16] Chinedu Wilfred Okonkwo and Abejide Ade-Ibijola. Chatbots applications in education: A systematic review. *Computers and Education: Artificial Intelligence*, 2:100033, 2021.

[17] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

[18] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897, 2020.

[19] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[20] Stanford HAI. How large language models will transform science, society, and ai, 2022. URL `https://hai.stanford.edu/news/how-large-language-models-will-transform-science-society-and-ai`.

[21] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *Advances in neural information processing systems*, 29, 2016.

[22] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. *arXiv preprint arXiv:1707.09457*, 2017.

[23] Zachary C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

[24] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of nlp models with checklist. *arXiv preprint arXiv:2005.04118*, 2020.

[25] Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. Defending against neural fake news. *Advances in neural information processing systems*, 32, 2019.

[26] Sarthak Jain and Byron C Wallace. Attention is not explanation. *arXiv preprint arXiv:1902.10186*, 2019.

[27] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019.

[28] Anna Jobin, Marcello Ienca, and Effy Vayena. The global landscape of ai ethics guidelines. *Nature machine intelligence*, 1(9):389–399, 2019.

[29] Che Jiang, Biqing Qi, Xiangyu Hong, Dayuan Fu, Yang Cheng, Fandong Meng, Mo Yu, Bowen Zhou, and Jie Zhou. On large language models' hallucination with regard to known facts. *arXiv preprint arXiv:2403.20009*, 2024.

[30] Razvan Azamfirei, Sapna R Kudchadkar, and James Fackler. Large language models and the perils of their hallucinations. *Critical Care*, 27(1):120, 2023.

[31] Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, et al. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*, 2021.

[32] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*, 2019.

[33] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[34] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[35] Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. Unifiedqa: Crossing format boundaries with a single qa system. *arXiv preprint arXiv:2005.00700*, 2020.

[36] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

[37] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR, 2020.

[38] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*, 2020.

[39] Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*, 2020.

[40] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623, 2021.

[41] Hila Gonen and Yoav Goldberg. Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them. *arXiv preprint arXiv:1903.03862*, 2019.

[42] Emily Dinan, Samuel Humeau, Bharath Chintagunta, and Jason Weston. Build it break it fix it for dialogue safety: Robustness from adversarial human attack. *arXiv preprint arXiv:1908.06083*, 2019.

[43] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR, 2022.

[44] Chih-Wei Kuo, Yueh-Fen Huang, and Hsiao-Ching Tsai. Adaptive query contextualization algorithm for enhanced information retrieval in alpaca llm. 2023.

[45] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[46] Luyu Gao, Zhuyun Dai, and Jamie Callan. Coil: Revisit exact lexical match in information retrieval with contextualized inverted list. *arXiv preprint arXiv:2104.07186*, 2021.

[47] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*, 2021.

[48] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[49] Rashid L Bashshur, Gary W Shannon, Noura Bashshur, and Peter M Yellowlees. The empirical evidence for telemedicine interventions in mental disorders. *Telemedicine and e-Health*, 22(2):87–113, 2016.

[50] CS Kruse, N Krowski, B Rodriguez, L Tran, J Vela, and M Brooks. Telehealth and patient satisfaction: a systematic review and narrative analysis. bmj open 7 (8): e016242, 2017.

[51] Saul N Weingart, David Rind, Zachary Tofias, and Daniel Z Sands. Who uses the patient internet portal? the patientsite experience. *Journal of the American Medical Informatics Association*, 13(1):91–95, 2006.

[52] Steven R Steinhubl, Evan D Muse, and Eric J Topol. The emerging field of mobile health. *Science translational medicine*, 7(283):283rv3–283rv3, 2015.

[53] Stefano Omboni, Tiziana Gazzola, Giorgia Carabelli, and Gianfranco Parati. Clinical usefulness and cost effectiveness of home blood pressure telemonitoring: meta-analysis of randomized controlled studies. *Journal of hypertension*, 31 (3):455–468, 2013.

[54] American Diabetes Association. 7. diabetes technology: standards of medical care in diabetes—2019. *Diabetes care*, 42(Supplement_1):S71–S80, 2019.

[55] Mitesh S Patel, David A Asch, and Kevin G Volpp. Wearable devices as facilitators, not drivers, of health behavior change. *Jama*, 313(5):459–460, 2015.

[56] Laura Dennison, Leanne Morrison, Gemma Conway, Lucy Yardley, et al. Opportunities and challenges for smartphone applications in supporting health behavior change: qualitative study. *Journal of medical Internet research*, 15(4):e2583, 2013.

[57] Jay Thakkar, Rahul Kurup, Tracey-Lea Laba, Karla Santo, Aravinda Thiagalingam, Anthony Rodgers, Mark Woodward, Julie Redfern, and Clara K Chow. Mobile telephone text messaging for medication adherence in chronic disease: a meta-analysis. *JAMA internal medicine*, 176(3):340–349, 2016.

[58] Artur Direito, Eliana Carraça, Jonathan Rawstorn, Robyn Whittaker, and Ralph Maddison. mhealth technologies to influence physical activity and sedentary behaviors: behavior change techniques, systematic review and meta-analysis of randomized controlled trials. *Annals of behavioral medicine*, 51(2):226–239, 2017.

[59] Rashid Bashshur, Charles R Doarn, Julio M Frenk, Joseph C Kvedar, and James O Woolliscroft. Telemedicine and the covid-19 pandemic, lessons for the future, 2020.

[60] N Murali[1] and N Sivakumaran. Artificial intelligence in healthcare–a review. 2018.

[61] Wei Luo, Dinh Phung, Truyen Tran, Sunil Gupta, Santu Rana, Chandan Karmakar, Alistair Shilton, John Yearwood, Nevenka Dimitrova, Tu Bao Ho, et al. Guidelines for developing and reporting machine learning predictive models in biomedical research: a multidisciplinary view. *Journal of medical Internet research*, 18(12):e323, 2016.

[62] Joseph A Cruz and David S Wishart. Applications of machine learning in cancer prediction and prognosis. *Cancer informatics*, 2:117693510600200030, 2006.

[63] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[64] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *nature*, 542(7639):115–118, 2017.

[65] Alvin Rajkomar, Jeffrey Dean, and Isaac Kohane. Machine learning in medicine. *New England Journal of Medicine*, 380(14):1347–1358, 2019.

[66] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

[67] Edward Choi, Andy Schuetz, Walter F Stewart, and Jimeng Sun. Using recurrent neural network models for early detection of heart failure onset. *Journal of the American Medical Informatics Association*, 24(2):361–370, 2017.

[68] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

[69] Eric J Topol. High-performance medicine: the convergence of human and artificial intelligence. *Nature medicine*, 25(1):44–56, 2019.

[70] Emma Beede, Elizabeth Baylor, Fred Hersch, Anna Iurchenko, Lauren Wilcox, Paisan Ruamviboonsuk, and Laura M Vardoulakis. A human-centered evaluation of a deep learning system deployed in clinics for the detection of diabetic retinopathy. In *Proceedings of the 2020 CHI conference on human factors in computing systems*, pages 1–12, 2020.

[71] Jonathan H Chen and Steven M Asch. Machine learning and prediction in medicine—beyond the peak of inflated expectations. *The New England journal of medicine*, 376(26):2507, 2017.

[72] Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and

Jeff Dean. A guide to deep learning in healthcare. *Nature medicine*, 25(1):24–29, 2019.

[73] Riccardo Miotto, Fei Wang, Shuang Wang, Xiaoqian Jiang, and Joel T Dudley. Deep learning for healthcare: review, opportunities and challenges. *Briefings in bioinformatics*, 19(6):1236–1246, 2018.

[74] Sebastian Gehrmann, Franck Dernoncourt, Yeran Li, Eric T Carlson, Joy T Wu, Jonathan Welt, John Foote Jr, Edward T Moseley, David W Grant, Patrick D Tyler, et al. Comparing deep learning and concept extraction based methods for patient phenotyping from clinical narratives. *PloS one*, 13(2):e0192360, 2018.

[75] Francesca Lake. Artificial intelligence in drug discovery: what is new, and what is next? In *Advances in Clinical Immunology, Medical Microbiology, COVID-19, and Big Data*, pages 539–545. Jenny Stanford Publishing, 2021.

[76] Jessica Vamathevan, Dominic Clark, Paul Czodrowski, Ian Dunham, Edgardo Ferran, George Lee, Bin Li, Anant Madabhushi, Parantu Shah, Michaela Spitzer, et al. Applications of machine learning in drug discovery and development. *Nature reviews Drug discovery*, 18(6):463–477, 2019.

[77] Feixiong Cheng, István A Kovács, and Albert-László Barabási. Network-based prediction of drug combinations. *Nature communications*, 10(1):1197, 2019.

[78] Assaf Gottlieb, Gideon Y Stein, Yoram Oron, Eytan Ruppin, and Roded Sharan. Indi: a computational framework for inferring drug interactions and their associated recommendations. *Molecular systems biology*, 8(1):592, 2012.

[79] WebMD. Webmd medical information dataset, 2024. URL `https://www.webmd.com/`. Accessed: 2024-03-17.

[80] Langchain-apidocs, . URL `https://api.python.langchain.com/`. Accessed: 2024-04-17.

[81] Google colaboratory. URL `https://colab.research.google.com`.

[82] Guido van Rossum and the Python development team. Python, 2024. URL `https://www.python.org/`.

[83] Thomas Wolf et al. Transformers: State-of-the-art natural language processing, 2024. URL https://huggingface.co/transformers.

[84] Hugging Face. Datasets, 2024. URL https://huggingface.co/docs/datasets.

[85] OpenAI. Openai api, 2024. URL https://beta.openai.com/docs/. Accessed: 2024-03-14.

[86] Apimedic. URL https://apimedic.com/. Accessed: 2024-02-03.

[87] Flutter. URL https://flutter.dev/. Accessed: 2024-02-11.

[88] Langchain, . URL https://langchain.com/. Accessed: 2024-02-17.

[89] Sebastián Ramírez. Fastapi, 2024. URL https://fastapi.tiangolo.com/.

[90] Uvicorn. URL https://www.uvicorn.org/.

[91] Daniel Gomes, Cristian Arango, and Carlos Abad. Pinecone: A scalable hybrid cloud storage system. *IEEE Transactions on Parallel and Distributed Systems*, 33 (7):1622–1636, 2022.