Kingdom of Saudi Arabia
Ministry of Education
Qassim University
College of Computer
Information Technology Dept.

المملكة العربية السعودية
وزارة التعليم
جامعة القصيم
كلية الحاسب
قسم تقنية المعلومات

Mureq : Agricultural Chatbot

Students:

Lamees AlOqlan 411201927

Rawf AlHarbi 411201943

Saja AlSaab 411201865

Supervisor:

Dr. Manal Alghieth

A project report submitted in partial fulfillment of the requirements for B.Sc. degree in Information Technology.

Qassim-Saudi Arabia

1445 (2023/2024)

Certificate

It is certified that the project report has been prepared and written under my direct supervision and guidance. The project report is approved for submission for its evaluation.

Dr. Manal Alghieth

Dedication

This work is dedicated to all those who have contributed to our educational journey, including those who have provided unwavering support, our families and friends who have been invaluable, and every student who shares our passion for learning.

Lamees AlOqlan , **Rawf AlHarbi** and **Saja AlSaab**

Acknowledgement

Above all, we extend our heartfelt gratitude to Allah for guiding and assisting us throughout the completion of this project. Furthermore, we would like to express our sincere appreciation to our project supervisor, Dr. Manal Alghieth, for her invaluable assistance in enhancing our skills and knowledge. We are also immensely grateful to all those who provided aid, support, and encouragement during our journey.

Lamees AlOqlan , **Rawf AlHarbi** and **Saja AlSaab**

Abstract

Nowadays, chatbots have become highly valuable tools for simplifying tasks across various fields. However, there is a noticeable lack of Arabic chatbots specifically tailored for the agricultural sector. This study aims to address this gap by exploring two transformer-based models, AraBERT v2 and QA AraBERT. These models are based on BERT (Bidirectional Encoder Representations from Transformers), a widely recognized and powerful transformer-based language model in the field of natural language processing (NLP). By leveraging the power of BERT, the chatbot will gain a deep understanding of the Arabic language and the contextual relationships between words. The BERT models underwent fine-tuning on a specialized Arabic dataset tailored to the agricultural sector. Focusing on the second model, QA AraBERT, the fine-tuning process yielded F1 score of 81.63%. This remarkable performance highlights the model's exceptional capability to grasp the nuances of agricultural inquiries. Empowered by this model, the chatbot, named Mureq, achieves high performance compared to other chatbots built with similar techniques. This success positions Mureq as a crucial role in supporting the realization of Saudi Arabia's 2030 vision for the agricultural sector, particularly in the realm of ornamental trees and plants.

Keywords: Chatbot, Arabic Language, Agriculture, Bidirectional Encoder Representations from Transformers, Natural Language Processing.

Table of Contents

List of Figures

List of Tables

Abbreviations

AI Artificial Intelligence

AIML Artificial Intelligence Markup Language

ALICE Artificial Linguistic Internet Computer Entity

ANN Artificial Neural Networks

API Application Programming Interface

BERT Bidirectional Encoder Representations from Transformers

BLEU Bilingual Evaluation Understudy

BRNN Bidirectional Recurrent Neural Networks

DL Deep Learning

DRL Deep Reinforcement Learning

EM Exact Match

FFNNs Feed Forward Neural Networsks

GLUE General Language Understanding Evaluation

GPT Generative Pre-trained Transformer

GPU Graphics Processing Unit

GRU Gated Recurrent Unit

GUI Graphical User Interface

JSON JavaScript Object Notation

LLMs Large Language Models

LSTM Long Short Term Memory

MAP Mean Average Precision

MCC Matthews Correlation Coefficient

ML Machine Learning

MLP Multi-layer Perceptron

MRR Mean Reciprocal Rank

NDCG Normalized Discounted Cumulative Gain

NLP Natural Language Processing

NMT Neural Machine Translation

QA Question-Answering

QAS Question Answering System

QLSTM Quaternion Long Short Term Memory

RAM Random Access Memory

RNNs Recurrent Neural Networks

SDLC System Development Life Cycle

Seq2Seq Sequence to Sequence model

T5 Text-to-Text Transfer Transformer

Chapter 1

INTRODUCTION

INTRODUCTION

1.1 Introduction

Chatbots, which are systems capable of engaging in natural language conversations with humans, have gained significant importance in recent years. Actually, this is attributed to the advancements in computational capabilities that enable more human-like interactions between computers and humans, approaching the level of interactions between humans themselves[1]. Consequently, there are more and more chatbots available that are intended to support and empower individuals by equipping them with information, fostering continuous learning, and enhancing their decision-making capabilities in diverse domains, including education, healthcare, customer service, personal productivity, finance, e-commerce, travel, agriculture and beyond.

In Saudi Arabia, agriculture faces challenges due to the arid climate and limited water resources. However, the country has recognized the importance of agriculture and its potential role in promoting environmental sustainability and addressing climate change. Therefore, it has taken significant steps towards fostering a green and sustainable agricultural sector. In line with this vision, the Saudi Green Initiative was launched. This comprehensive program aims to promote environmental conservation, reduce harmful emissions, and ensure the sustainability of natural resources. One of the aspects that the initiative focuses on is the Sustainable Agriculture Promotion and Biodiversity Conservation, which involves tree planting , sustainable land use and the preservation of natural areas and reserves[2].

By leveraging the advancements in technology, chatbots have the potential to revolutionize the agricultural sector in Saudi Arabia. They can provide valuable support, knowledge, and guidance to farmers, gardeners, and people interested in planting , empowering them with the knowledge needed to make accurate and informed decisions.

1.2 Chatbot

This section offers a comprehensive exploration of chatbots, focusing on their definition, the underlying architecture that powers them, the various types that exist, and the techniques employed in their development process.

1.2.1 Definition

The integration of Artificial Intelligence (AI) has progressively permeated various aspects of our daily lives, encompassing the creation and evaluation of intelligent software and hardware known as intelligent agents. These agents exhibit diverse capabilities, spanning from manual labor to intricate operations. Among these AI systems, chatbots stand out as a prominent example and a common illustration of intelligent interaction between humans and computers [3]. It is a computer program, which responds like a smart entity when conversed with through text or voice and understands one or more human languages by Natural Language Processing (NLP)[4]. In the lexicon, a chatbot is defined as "A computer program designed to simulate conversation with human users, especially over the Internet" chatbots are also known as smart bots, interactive agents, digital assistants, or artificial conversation entities.

1.2.2 Architecture

Chatbot architecture typically comprises five primary elements: the user interface, Natural Language Processing (NLP), dialog management, data storage, and a knowledge base, as shown in Figure1.1. The user interface enables users to interact with the chatbot, while NLP processes and comprehends user inputs, including intent recognition and entity extraction. Dialog management maintains conversation context, controls the flow of the conversation, and decides suitable responses. The knowledge base stores the information and data needed for the chatbot to provide accurate and relevant answers. Additionally, chatbot architectures often include a data storage component for interaction history and analytics. This component stores and manages conversation logs and user data, allowing the chatbot to personalize responses based on user history and preferences. It also facilitates analysis to improve the chatbot's performance and understand user behavior patterns. Together, these components work in harmony to enable effective communication and interaction between the chatbot and user.

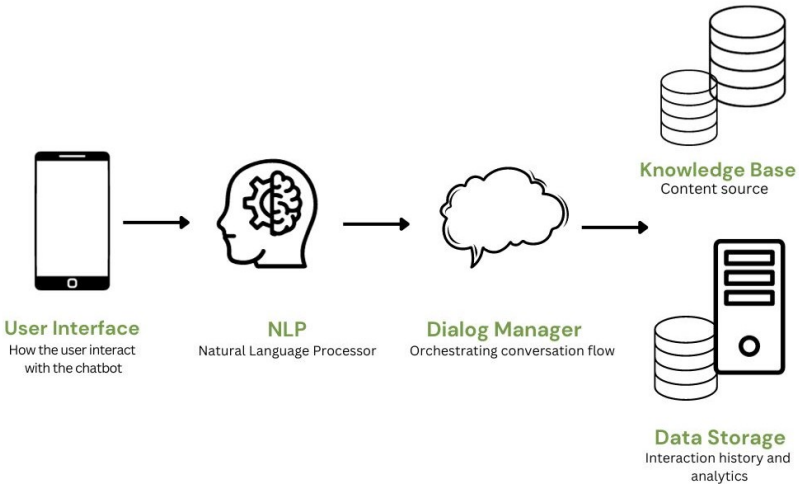


Figure 1.1: Chatbot architecture.

1.2.3 Types

Chatbots, the computer programs designed to simulate human conversation, can be classified based on various parameters that define their characteristics and functionalities. These parameters help categorize chatbots into different types, enabling us to understand their purposes, capabilities, and limitations. Here are some parameters that can be used to classify chatbots:

- Knowledge Domain

  In this context, chatbots are categorized according to the knowledge they have access to or the extent of their training data.

  1. Open Domain

     Open-domain chatbots are supposed to converse freely with humans without being restricted to a topic, task or domain[5]. These chatbots aim to simulate human-like conversation across a wide range of subjects. These chatbots leverage large-scale language models and advanced Natural Language Processing (NLP) techniques to understand user input, generate contextually relevant responses, and maintain coherent and engaging interactions. By being unconstrained by predefined domains, open-domain chatbots have the potential to provide more versatile and dynamic conversational experiences, catering to a diverse array of user interests and inquiries.

2. Closed Domain

   Closed-domain chatbots are chatbots that operate within specific domains or areas of expertise, focusing on providing targeted assistance or information in a limited subject area. These chatbots are designed to offer specialized and accurate responses by being trained on a specific set of knowledge or data related to their designated domain. For example, a closed-domain chatbot in the healthcare field may be programmed to provide information about symptoms, treatments, or general medical advice. The advantage of closed-domain chatbots lies in their ability to offer precise and reliable information within their specialized domain, leading to enhanced user experiences.

- Goal

Classification based on the goals considers the primary goal that chatbots aim to achieve.

1. Informative

   An informative chatbots are the chatbots that specifically designed to provide information and answer user queries on a wide range of topics. Its primary function is to deliver accurate and relevant information to users in a conversational manner. They actually employ NLP techniques to understand user questions or prompts and retrieve information from a knowledge base or database. They can cover various domains, such as general knowledge, news, weather, sports, entertainment, and more. These chatbots are trained on large datasets or have access to reliable sources of information to ensure the accuracy of their responses.

2. Conversational

   These chatbots engage in conversations with users, aiming to mimic human-like interactions. Their primary objective is to provide accurate responses based on the input they receive from the user. As a result, they are often developed with the intention of sustaining a continuous dialogue with the user [6].

3. Task-based

   A task-based chatbots are designed to perform specific tasks or actions based on user requests or commands. These chatbots have a more defined purpose and are optimized for completing particular tasks efficiently. The main goal of a task-based chatbot is to assist users in accomplishing specific actions. These tasks can vary widely depending on the domain or application of the chatbot. Examples of tasks that task-based chatbots can handle include making reservations, providing product recommendations, answering frequently asked questions, processing transactions, and more [6].

- Service Provided Input Processing and Response Generation Method

  Classification based on input processing and response generation methods in chatbots considers how the chatbot handles user inputs and generates appropriate responses. This classification helps understand the underlying mechanisms and techniques used by chatbots to interact with users effectively[7]. There are three models used to produce the appropriate responses:

  1. Rule-based Model

     A rule-based model chatbot is a type of architecture which most of the first chatbots have been built with, like numerous online chatbots. They choose the system response based on a fixed predefined set of rules, based on recognizing the lexical form of the input text without creating any new text answers. The knowledge used in the chatbot is humanly hand-coded and is organized and presented with conversational patterns. A more comprehensive rule database allows the chatbot to reply to more types of user input. However, this type of model is not robust to spelling and grammatical mistakes in user input [7].

  2. Retrieval-based Model

     A retrieval-based model is a type of chatbot model that generates responses by retrieving pre-existing responses from a pool of options rather than generating them from scratch. This approach leverages a collection of predefined responses, typically stored in a knowledge base or indexed resources. In a retrieval-based model, the process begins with data collection and indexing. Conversations or a knowledge base

containing pairs of user inputs and corresponding responses are used to train the model and create an index of the responses for efficient retrieval.

3. Generative Model

   A generative-based model is a type of chatbot model that generates responses from scratch rather than retrieving pre-existing responses. It utilizes NLP and Machine Learning (ML) techniques to generate contextually relevant and coherent responses based on the input received. In a generative-based model, the training process involves exposing the model to a large dataset of conversations or text corpora. The model learns the underlying patterns, language structures, and context from the training data. This allows it to generate responses that are not limited to a predefined set of options[7].

1.2.4 Techniques Used

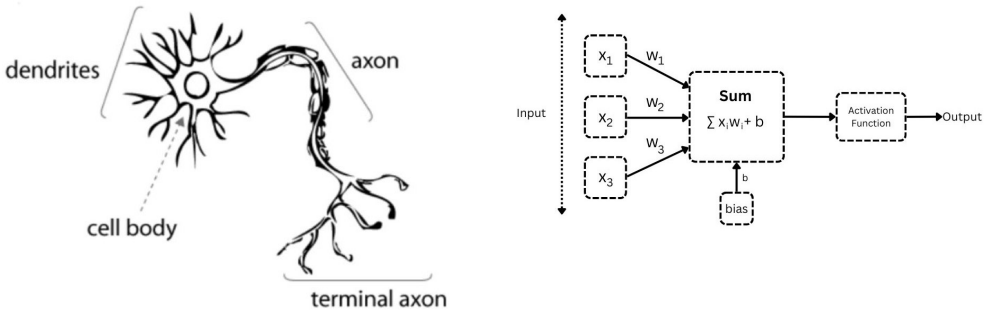
Artificial Intelligence (AI) techniques have revolutionized the field of chatbots development, providing the backbone for more intelligent, responsive, and user-friendly interactions. Chatbots, designed to emulate human conversation, leverage these AI techniques to better understand and respond to user queries, adapt from past interactions, and offer more personalized responses. Key AI techniques used in chatbot development include Machine Learning (ML) , Natural Language Processing (NLP), Deep Learning (DL) and Large Language Models (LLMs).

1.2.4.1 Natural Language Processing

Natural Language Processing (NLP) is an area of artificial intelligence that focuses on the interaction between computers and human language. Its purpose is to develop algorithms and methods that allow machines to understand, interpret, and generate meaningful text or speech. NLP encompasses various tasks such as speech recognition, language translation, sentiment analysis, and text classification. By utilizing techniques from linguistics, statistics, and machine learning, NLP algorithms can process and analyze large amounts of textual or spoken data. The ultimate goal of NLP is to bridge the gap between human language and computer comprehension, enabling applications like virtual assistants, chatbots, and information retrieval systems.

1.2.4.2 Machine Learning

Machine Learning (ML) is a branch of artificial intelligence that focuses on the development of algorithms and models that enable computers to learn from data and make predictions or decisions without being explicitly programmed for each task. ML algorithms allow machines to automatically identify patterns, extract insights, and improve their performance through experience. One of the key techniques used in ML is Artificial Neural Networks (ANN). ANN is a computational model inspired by the structure and functioning of the human brain. It consists of interconnected artificial neurons that process and transmit information, as shown in Figure 1.2. ANN algorithms are capable of learning complex relationships and patterns in data, making them well-suited for various tasks.



(a) Human Neurons [8] (b) Artificial Neurons

Figure 1.2: Human and artificial neurons.

1.2.4.3 Deep Learning

Deep Learning (DL) is a subset of Machine Learning (ML) that focuses on training Artificial Neural Networks (ANN) with multiple layers to learn and make sense of complex patterns and data representations. DL algorithms enable computers to automatically learn and extract hierarchical representations from large amounts of data, leading to advancements in various fields such as computer vision, Natural Language Processing (NLP), and speech recognition. DL models, known as deep neural networks, consist of numerous layers of interconnected artificial neurons that progressively extract higher-level features and abstractions from the input data. The ability of DL models to automatically learn and generalize from data makes them powerful tools for solving complex problems, including image and speech recognition, autonomous driving, drug discovery, and more.

DL techniques have transformed the landscape of Artificial Intelligence (AI), enabling computers to learn and interpret complex patterns and representations. Among the most prominent DL techniques are Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Generative Adversarial Networks (GANs), Transformers, and Autoencoders. CNNs have excelled in computer vision tasks by automatically extracting spatial features from images and videos. In contrast, RNNs have been highly effective in sequential data processing, particularly for NLP and speech recognition. GANs, on the other hand, have gained widespread acclaim for their ability to generate realistic data samples. Transformers have brought about a significant shift in Natural Language Understanding (NLU), capturing contextual relationships between words in a sequence. Lastly, autoencoders have found applications in unsupervised learning and data compression. Together, these DL techniques have propelled advancements in computer vision, NLP, and data generation, redefining the possibilities of AI.

1.2.4.4 Large Language Model

The Large Language Models (LLMs) are types of artificial intelligent algorithms that use transformer model as a deep learning technique. These models have undergone extensive training on vast quantities of text data to facilitate their comprehension and generation of text that closely mimics human language. As a result of this massive training, they have earned the designation of "large language models". They consist of billions of parameters and can perform a wide range of language-related tasks. These tasks encompass various domains such as text generation, language translation, text summarization, sentiment analysis, question answering, and many other domains. Significantly, they are utilized in chatbots and virtual assistants to facilitate more natural and interactive conversations.

1.3 Problem Specification and Motivation

There are several notable challenges in Saudi Arabia that hinder the successful growth of plants. One of the challenges is the scarcity of information that assists people in planting and caring for plants. This lack of knowledge can lead to unsuccessful planting endeavors, resulting in the wastage of resources and a diminished environmental yield. Another challenge for Arabic-speaking farmers and gardeners is the lack of Arabic language support in plant cultivation and related technology.

Addressing these issues is motivated by Saudi Arabia Vision 2030's ambition to improve the environment through plant cultivation, indicating a commitment to guaranteeing a more sustainable and greener future for coming generations. Nonetheless, there are several factors that are also considered as a motivation for choosing chatbot as the desired application of the project, such as the continuous operation without time constraints , the user-friendly interface and efficient communication with users. These factors contributed to it suitability for the intended project.

1.4 Goals and Objectives

The goal behind the project is to create a chatbot that will improve the agriculture field in Saudi Arabia while the objectives of the project can be summarized as follows:

1. Examine prior research to determine the most effective methods and existing gaps.
2. Gather relevant data for the research that is appropriate for the addressed location.
3. Design a proposed methodology that covers the noted gap and considers the characteristics of the information gathered.
4. Develop the chatbot based on the designed methodology.
5. Test the developed chatbot and assess the outcome.
6. If applicable, incorporate the chatbot with other platforms.

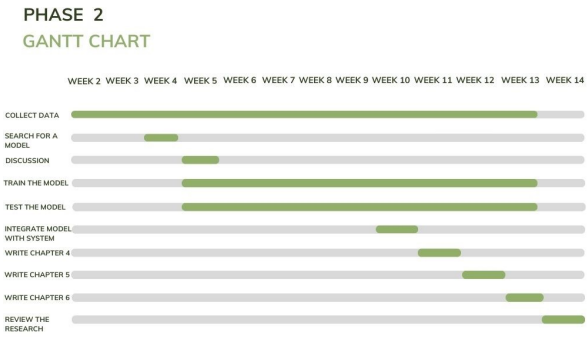
1.5 Study Scope

This project focuses on the development of an AI-powered chatbot system specifically tailored to overcome challenges of planting ornamental plants and trees in Saudi Arabia. The chatbot will serve as a valuable resource for farmers, gardeners, and individuals interested in planting ornamental plants and trees by offering localized knowledge, guidance, and support.

1.6 Study Plan and Schedule

The charts in Figure 1.3 offer clear overviews of the project timeline, tasks, and their respective durations for both Phase 1 and Phase 2.

(a) Phase 1 plan



(b) Phase 2 plan

Figure 1.3: The project workflow plan

1.7 Organizing of the Chapters

The project is divided into two phases, each consisting of three chapters, and the chapters as following:

Chapter 1 (Introduction): Provides an overview of the project, including comprehensive exploration of chatbots, problem specification and motivation, goals and objectives, project scope, and project plan.

Chapter 2 (Literature Review): Provides a historical overview of chatbots, reviews previous studies and discusses their findings, addresses current issues, and propose future work in the field.

Chapter 3 (Methodology): Provides a detailed description of the type of study, methodology approach, and proposed methodology including proposed framework, requirements, data collection, data structure, and system design procedure consisting of the conversation flow design and proposed methodology design.

Chapter 4 (Implementation and Testing): provides steps on how to implement the model, which includes dataset preparation and model training, model evaluation, system architecture, and its components, as well as testing and the metrics used for testing.

Chapter 5 (Result and Discussion): outlines the obtained results and major findings, followed by an in-depth analysis and interpretation.

Chapter 6 (Conclusion and Future work): covers the conclusion, contributions, and implications of the study, limitations, and future work.

Chapter 2

LITERATURE REVIEW

LITERATURE REVIEW

2.1 Introduction

This chapter gives a thorough overview of the history of chatbot development. It outlines important turning points and innovations in chatbot technology, following its beginnings from the early days to the present. Furthermore, the chapter delves into several relevant studies and presents their conclusions to offer an in-depth review as well as identify issues related to the current studies.

2.2 History of Chatbot

The development of chatbot technology has had a remarkable journey through various stages, each of which has further developed conversational AI.

It all began with the early days of pattern matching, when programmed responses were generated by chatbots by matching user input with pre-defined patterns. ELIZA was a famous chatbot in the 1960s designed to mimic a psychotherapist [9]. It made use of pattern matching and response selection systems built around templates. ELIZA's limited knowledge was one of its shortcomings, as it hindered its capacity to participate in discussions on a variety of subjects. However, ELIZA was an inspiration for later chatbots.

The development of chatbots reached a turning point with Parry's invention in the 1970s, which demonstrated the ability of Natural Language Processing (NLP) by using pattern matching in conjunction with an expert system to convincingly mimic the delusions, suspicion, and disorganized thinking that mark out those who suffer from paranoid schizophrenia [10].

Then, Artificial Intelligence (AI) was first used in chatbots when Jabberwacky was created in 1988 [11]. Jabberwacky employed contextual pattern matching to answer based on past chats and was created in CleverScript, a language based on spreadsheets that aided the construction of chatbots. However, Jabberwacky is unable to respond quickly or handle a large number of users.

In 1995, another milestone in the history of chatbots was the development of the Artificial Linguistic Internet Computer Entity (ALICE), the first online chatbot inspired by ELIZA [12]. With no real understanding of the entire dialogue, ALICE was built on pattern matching. It had a web-based discussion feature that permitted longitude and covered any subject. The key distinction between ALICE and ELIZA is that ALICE was constructed using Artificial Intelligence Markup Language (AIML), a new language established specifically for this purpose.

When SmarterChild was developed in 2001 and made available on messengers like Microsoft Network (MSN) and America Online (AOL)[13], chatbot technology truly advanced. For the first time, a chatbot that could retrieve information from many databases was able to assist people with everyday chores.

The development of intelligent chatbots continued with the creation of intelligent voice assistants, integrated into smartphones or dedicated home speakers that can understand voice commands and handle tasks such as home automation device monitoring, calendar, e-mail, and Others. Siri from Apple was the first in this field and was released as a standalone app in 2010 and added to iOS in 2011. Microsoft Cortana followed shortly in 2013. Amazon released Alexa in 2014, and Google Assistant was announced in 2016 in its home speakers and was also included in Google apps for smartphones based on Android[14].

The history of chatbots advanced significantly with the development of NLP, specifically with the appearance of word embeddings [15]. Word embedding is a technique for putting words with similar meanings close to one another in a continuous vector space, where words or phrases are represented as dense vectors. Word embeddings can help chatbots better understand and produce human language by incorporating them into a wide range of NLP tasks. Words are encoded into dense vector representations by word embedding models like Word2Vec, GloVe, and fastText using neural network architectures.

Recurrent Neural Networks (RNNs) is a sort of Artificial Neural Network (ANN) that is widely employed in the construction of chatbots. When processing input data, RNNs take into account the previous steps in the sequence as well as the current step. For chatbots, this means that the network may take into consideration the earlier messages and responses in a discussion, enabling it to produce more contextually relevant and coherent replies. In

RNN-based chatbots, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are two common recurrent unit types. The LSTM units are intended to alleviate the vanishing gradient problem that can occur when training deep neural networks, such as RNN. Long-term dependencies can be captured more efficiently using LSTM units since they have a memory cell that can store information over lengthy sequences. As opposed to LSTM units, GRU units are a more simplified form of LSTM units. They are designed to achieve comparable performance with fewer parameters, which increases their computational efficiency during training and usage [16].

The Sequence-to-Sequence (Seq2Seq) model is a fundamental machine translation technique that can also be applied to chatbots. A basic Seq2seq model consists of two RNNs, an encoder and a decoder. The input sequence, such as a user's question or statement, is processed by the encoder and mapped into a context vector, a fixed-length vector representation. The input sequence is compressed into this vector, which is then given to the decoder to produce the proper answer[17]. RNNs, such as LSTM or GRU, that can process sequential data can be used to create the encoder and decoder.

In recent years, NLP and chatbots have been revolutionized by transformers. The transformer conception was initially presented in Vaswani et al.'s paper "Attention is All You Need" in 2017[18]. Although it was first created for machine translation tasks, it has subsequently been used for many other NLP tasks, such as chatbots. Transformers are notable for their self-attention mechanism, which allows the model to interpret an input sequence while focusing on different segments of it.

Later on, the development of chatbots became more advanced thanks to the appearance of Large Language Models (LLMs) that utilize the transformer architecture. LLMs Enabled transfer learning by introducing model fine-tuning to a number of LLMs such as Generative Pre-trained Transformer (GPT), Bidirectional Encoder Representations from Transformers (BERT), Text-to-Text Transfer Transformer (T5), and many more.

More sophisticated and engaging conversational agents have been made possible by the development of chatbots from pattern matching to LLMs. Every phase has helped create chatbots that are capable of carrying out meaningful and context-aware dialogues, offering significant support and help across a range of industries.

2.3 Related Studies

This section discusses previous studies on chatbots and their technologies, highlighting the advancements that have been achieved in this field. By reviewing these previous works, we can gain a comprehensive understanding of the research landscape surrounding chatbots and identify key areas for further investigation.

One of the widely utilized techniques, both historically and presently, is the application of Recurrent Neural Networks (RNNs). RNNs prove to be particularly valuable in scenarios where sequential data and temporal dependencies play a crucial role. It excels in processing variable-length sequences and is especially pertinent in situations where interpretability and customization hold significant importance. The study [19] investigated the creation of a chatbot specifically designed for the agricultural sector in India, employing advanced techniques such as Natural Language Processing (NLP) and Deep Learning (DL). Specifically, it employed Multilayer Perceptron (MLP) and RNNs models, with the latter achieving an impressive accuracy score of 97.83% and the former achieving 96.97%. Additionally, the study [20] introduced a deep learning model for a conversational chatbot using the Tensorflow software library, particularly the Neural Machine Translation (NMT) model. The proposed work employed Bidirectional Recurrent Neural Networks (BRNN) with attention layers to handle extensive text inputs.

Long Short-Term Memory (LSTM) is a powerful tool for modeling sequential data and has demonstrated its effectiveness in various NLP tasks. Its ability to capture long-term dependencies makes it well-suited for improving Question Answering Systems (QAS) by enhancing comprehension, context understanding, and information retrieval. The authors in [21] presented a QAS for natural science questions, focusing on improving answer selection by utilizing LSTM deep learning model to overcome limitations in manual feature engineering in Indonesia. The model was trained using a dataset of question-answer pairs from scientific articles. The performance was assessed using Mean Average Precision (MAP) Mean Reciprocal Rankand (MRR) metrics and was improved with smaller dropout rates, 50 hidden units, lower learning rates, a margin of 0.1, and a smaller answer pool size. Therefore, the final model with optimized parameters achieved impressive results, with an MRR of 90.06% and a MAP of 78.69%. As well as the authors in [22] discussed the challenges and solutions in developing Arabic open-domain

QAS. A deep learning technique was proposed for an open-domain QAS, which included data pre-processing, name entity relationship, and response retrieval. The QAS was constructed using the Multinomial Naïve Bayes (MNB) algorithm and the Embedding from Language Models approach with a Quaternion Long Short Term Memory (QLSTM) neural network. In fact, it achieved high accuracy, precision, recall, F1-score, MCC, and Kappa when evaluated on the Arabic reading comprehension dataset (ARCD) and TyDiQA dataset. Moreover, the study [23] discussed KisanQRS, a query-response system designed to assist Indian farmers. It used a threshold-based clustering algorithm to group similar queries together, and a LSTM. Furthermore, it employed a rule-based approach to filter clustered queries and select the most appropriate top answers. Lastly, the system has been evaluated and found to outperform traditional techniques, with the query mapping module achieving a top F1-score of 96.58% and the answer retrieval module achieving a competitive Normalized Discounted Cumulative Gain (NDCG) score of 96.20% on a 10,000 sample size.

Attention mechanisms are crucial elements in contemporary deep learning models, especially in the domains of NLP and computer vision. In the field of NLP, attention mechanisms play in various tasks including machine translation, question-answering, and text summarization, these mechanisms aid in aligning and focusing on significant words or phrases within the input sequence, enabling the model to capture the interdependencies and connections between different sections of the text. Therefore, the performance of QAS can be improved by employing attention mechanisms. The study [24] focused on developing a chatbot using a Seq2Seq model, which was trained on a university admission dataset. The model, evaluated on a small dataset, produced a high Bilingual Evaluation Understudy (BLEU) score of 41.04. An attention mechanism technique using reversed sentences improved the model to a higher BLEU of 44.68. Similarly, authors in [25] examined a conversational chat system that employed an attention mechanism to address COVID-19 inquiries. The system incorporated encoder-decoder architecture with a Gated Recurrent Unit (GRU) with a decoder that utilized bidirectional GRU and employed the attention mechanism based on the Luong attention mechanism. Actually, this mechanism encompassed three scoring methodologies: dot, general, and concat. Subsequently, the study conducted a performance comparison of these attention mechanisms and concluded that the dot attention mechanism achieved the highest accuracy, reaching 87%. Ultimately, it is proven that the integration of the attention mechanism significantly enhanced chatbot's accuracy in

comprehending and responding to user inquiries, in contrast to scenarios where the attention mechanism was not employed. Moreover, the study [26] presented methods for implementing deep learning-based symbolic processing in QAS. These methods aimed to overcome the limitations of conventional techniques when dealing with vast amounts of data and unfamiliar symbols. The proposed techniques enable the processing of inputs containing unknown symbols, efficient learning with limited training data, and the generation of comprehensive representations through the utilization of deep learning models in the field of NLP such as Word2Vec algorithm and the attention mechanism.

Deep Reinforcement Learning (DRL) plays a crucial role in the development and enhancement of chatbots. By leveraging deep neural networks and reinforcement learning algorithms, chatbots can learn to optimize their responses based on user interactions and feedback. DRL enables chatbots to handle complex dialogue scenarios, understand user intents, and generate contextually relevant replies. It empowers chatbots to dynamically generate responses, personalize interactions, and balance exploration and exploitation to improve their conversational abilities. Furthermore, DRL allows chatbots to optimize for long-term rewards, consider the overall conversation flow, and continuously adapt and improve over time. With these capabilities, chatbots trained with DRL can provide more interactive, engaging, and effective conversational experiences for users. The study [27] presented a DRL chatbot called Milabot which was developed by the Montreal Institute for Learning Algorithms (Mila). Milabot is able to converse with humans on common small talk topics through speech and text. The system consists of a set of Natural Language Generation (NLG) and retrieval models, including neural network and template-based models. By applying reinforcement learning to collective data and real-world user interactions, the system was trained to select the appropriate response from the existing models in its collection. The system was evaluated through A/B testing with real-world users, where it performed much better than other systems. The results highlight the potential of pairing group systems with DRL as a fruitful path for the development of real-world, open-field conversational agents. Additionally the study [28] introduced an ensemble-based DRL approach for training chatbots to engage in human-like conversations. In order to address the challenge of developing chatbots that exhibit fluent and coherent dialogue, the authors proposed a novel reward function for evaluating chatbot performance. This reward function demonstrates a strong correlation with human judgments and proves effective in measuring the fluency, engagingness,

and consistency of dialogues. Moreover, the experimental results are supported by both quantitative evaluations using held-out data and a human evaluation that assesses the quality of dialogues. In conclusion, the findings showcased the potential of the proposed ensemble-based DRL approach in training chatbots capable of engaging in human-like conversations.

While RNNs have been widely used in chatbot development, transformers have emerged as a powerful alternative due to their ability to capture long-range dependencies and model contextual information effectively. Transformers can generate more coherent and contextually relevant responses, leading to improved conversational quality and user experience. The study [29] introduced AraBERT, a transformer-based model specifically designed for Arabic language understanding. By pre-training on a large corpus of Arabic text and evaluating its performance against other models, the authors demonstrate that AraBERT achieved state-of-the-art results across various Arabic NLP tasks. Additionally, authors in study [30] focused on developing an Arabic healthcare assistant chatbot. By experimenting with BERT models, alongside Logistic Regression with TF-IDF and Doc2Vec, the authors evaluated their performance on an Arabic dataset. Finally, the results indicated that the BERT model achieved the highest accuracy rate of 95%, outperforming the other models. The study [31] conducted controlled experiments to assess the performance of pre-trained transformer models focusing specifically on BERT and its variations (RoBERTa, DistillBERT, XLNet, and ALBERT) on real-world datasets and benchmarks. The results align with previous literature, with RoBERT consistently performing better on text classification, sequence tagging, question answering, and the General Language Understanding Evaluation (GLUE) benchmark. XLNet performs well in some tasks but shows lower performance on the GLUE benchmark and individual datasets like CoLA. BERT demonstrates median performance across tasks, while ALBERT, despite being smaller, achieved the best performance. On the other hand, the study [32] focused on improving the accuracy and performance of a Bengali chatbot by applying the transformer model. Ultimately, it employed a Bengali general knowledge question answer dataset to evaluate and compare the performance of the transformer model and a Seq2Seq model with attention. Actually, the transformer model achieved a significantly higher BLEU score of 85.0 on the applied question-answer data, indicating its superior performance. In contrast, the Seq2Seq model with attention achieved a BLEU score of 23.5, highlighting the limitations of this approach. Furthermore, the study [33] made a comparison of the same models, the Seq2Seq with attention mechanism and the transformer.

The findings suggested that the transformer model outperformed the Seq2Seq model in terms of training efficiency. Additionally, training transformer networks was found to be easier than training LSTM networks due to the smaller number of parameters involved. Moreover, the transformer model demonstrated faster performance compared to models based on RNNs.

The Table 2.1 summarizes the results of the previously discussed studies showcasing techniques, dataset source, evaluation metrics, and accuracy focusing more on question answering tasks in some papers.

Ref.	**Techniques**	**Data source**	**Evaluation metric**	**Accuracy (%)**
[19], 2021	RNN and MLP	From World Repository	Accuracy	RNN achieved 97.83 while MLP 96.97
[20], 2020	BRNN with an attention mechanism	From Reddit	BLEU	30.16
[21], 2021	A combination of the Word2Vec model with the biLSTM model.	From Wikipedia	Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR)	MAP achieved 90.06 while MRR 78.69

Table 2.1 continued from previous page

[22], 2022	Embeddings from ELMo, Multi Nominal Naïve Bayes(MNB) and QLSTM.	Arabic reading comrehension dataset (ARCD)	Accurarcy Precision recall F1 score MCC Kappa	Acc=96.23 Prec =97 Rec= 96.95 F1=97 MCC=95.98 Kappa=95.7
		TYDIQA		Acc =95.35 Prec=94.8 Rec=94.68 F1=94.73 MCC=92.98 Kappa=93.6
[23], 2023	LSTM With SBERT	Kisan Call Center (KCC) consists of 34 million call logs	F-1 score	96.58
[24], 2019	Encoder-decoder with biLSTM combined with attention mechanism,and reversed words.	From Telkom university addmission	BLEU	44.68
[25], 2023	Encoder-decoder with Gated Recurrent Unit (GRU) combined with attention mechanism	TransPerfect	BLEU	87
[26], 2019	Encoder-decoder with LSTM combined with attention mechanism and Word2Vec model	From Kinsources and Geoquery	A correct answer rate in QAS	From 61.4 to 100

Table 2.1 continued from previous page

[27], 2018	Deep Reinforcement Learning	Amazon Mechanical Turk	User score	3.15 out of 5
[28], 2019	Ensemble Deep Reinforcement Learning	Persona-Chat dataset	Automatic evaluation and human evalution	Not clearly mentioned
[29], 2021	Pre-trained BERT	Arabic Corpus, OSIAN and SquAD for QA task	Sentence Match for QA task	93
[30], 2021	Pre-trained BERT	-	Accuracy	95
[31], 2022	Pre-trained transformer models	SquAD for QA task	F-1 score and Exact Match(EM) for QA task	RoBERT achieved the highest F1 and EM of 91.82 and 85.42
[32], 2022	Seq2Seq with attention mechanism vs transformer model.	Bengali general knowledge Question Answer dataset.	BLEU	Seq2Seq with attention mechanism 23.5 while transformer achieved 85.
[33], 2023	Seq2Seq with attention mechanism vs transformer model.	Cornell movie dialogue corpus	-	-

Table 2.1: Summary of previous studies

2.4 Proposed Work

Based on the previous studies, it can be concluded that both RNNs and DRL technologies have shown promising results in chatbot development. However, transformer models have emerged as a superior alternative, surpassing the performance of RNNs and DRL-based models in terms of generating coherent and contextually relevant responses. Additionally, the studies indicate that there is a lack of Arabic chatbots specifically designed for the agricultural field. To address this gap, leveraging transformer models could be a potential approach to develop Arabic chatbots that cater to the agricultural domain. These transformer models have the capability to capture long-range dependencies and effectively model contextual information, enabling them to provide advanced conversational experiences and valuable assistance to users in the agricultural sector.

2.5 Issues Related to the Current Work

Based on our analysis of previous studies, it can be concluded that there are significant challenges and issues currently faced in the development and implementation of chatbot systems within the agricultural sector. This section highlights the most prominent, which are as follows:

1. Developing chatbots that can accurately understand and interpret natural language is a challenging task. This is particularly true when dealing with complex or ambiguous language, such as Arabic misinterpretations and irrelevant responses can result in a frustrating user experience.

2. The absence of emotional intelligence poses a notable challenge in the development of chatbots, as they frequently encounter difficulties in understanding and responding suitably to users' emotions, tone, or sarcasm. This constraint can result in interactions that appear mechanical and impersonal, ultimately falling short of users' desires for more empathetic and human-like engagements.

3. Building a successful chatbot relying on plants that grow in Saudi Arabia can be difficult due to limited availability, incompleteness, or outdatedness of data. This lack poses a significant challenge in training the chatbot effectively, impacting its accuracy and reliability.

4. Developing agricultural chatbots requires seamless integration with third-party applications, such as weather forecasting, irrigation systems, or sensors. However, this integration can be complex and resource-intensive, posing a significant challenge.

2.6 Summary

The chapter provided a thorough history of the evolution of chatbot technology, starting with pattern matching and concluding with LLMs. It also identified the research gap, cited a number of notable studies, provided a summary of their conclusions in Table 2.1, and listed some notable issues related to the current works.

Chapter 3

METHODOLOGY

METHODOLOGY

3.1 Introduction

This chapter will provide a general overview of the study type, methodology approach, framework, data used and System Design to create the planned work effectively. The goal of this chapter is to evaluate and explain the methods employed by this study to achieve its goal of developing an Arabic agricultural chatbots using deep learning techniques.

3.2 Type of Study

This study is a combination of theoretical and experimental approaches. It begins by establishing a conceptual framework that outlines the essential components, functionalities, and objectives of the Arabic agricultural chatbot. This framework provides a theoretical basis for the development process. Furthermore, a comprehensive literature review is conducted to explore existing research, theories, and models pertaining to chatbot technology. This review helps identify gaps, best practices, and potential challenges, contributing to the effective development of the Arabic agricultural chatbot.

The experimental side of the study focuses on the practical implementation of the Arabic agricultural chatbot, building upon the theoretical framework. This involves coding and programming. Additionally, real-world data relevant to ornamental plants and trees such as how to cultivate, care and optimal climate conditions for its cultivation, is collected and incorporated into the chatbot. This aids in providing accurate and valuable information to users.

3.3 Methodology Approach

Figure 3.1 below illustrates the System Development Life Cycle (SDLC) model, which comprises six distinct phases of project management: planning, analysis, design, implementation, testing.

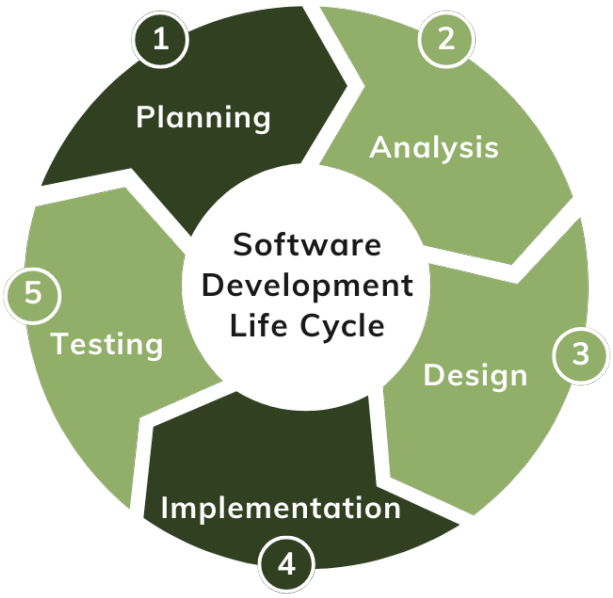


Figure 3.1: System Development Life Cycle

Planning: In the planning phase, the Arabic agricultural chatbot goals were established. This included gaining an understanding of the target audience, identifying their requirements, and determining the necessary functionalities and features of the chatbot. Furthermore, the project scope and timeline were defined.

Analysis: In the analysis phase, a comprehensive understanding of the requirements for the Arabic agricultural chatbot was achieved. This involved gathering and analyzing detailed requirements for the chatbot functionality, capabilities, and integrations.

Design: During the design phase, the chatbot's architecture and design are carefully planned. This includes defining the chatbot's conversational flow, and proposed methodology design.

Implementation: In the upcoming implementation phase, the chatbot will be developed based on the design specifications. We will write the code, integrate necessary datasets, and implement the deep learning algorithms required for understanding and generating responses. Iterative development and testing will be performed to ensure the functionality and quality of the chatbot.

Testing: In the upcoming testing phase, various testing techniques will be employed to validate the performance of the chatbot. Testing will encompass validating the chatbots accuracy in providing agricultural information.

3.4 Proposed Framework

In this study, BERT, a transformer-based model, will be employed as the core framework for developing an Arabic agricultural chatbot called Mureq, as shown in Figure 3.2. BERT, short for Bidirectional Encoder Representations from Transformers, is a cutting-edge language model in the field of Natural Language Processing (NLP). With its transformer architecture, BERT can effectively capture contextual information and dependencies within text data. By utilizing BERT's advanced language understanding capabilities, we aim to enhance the chatbot's ability to comprehend and generate responses that are contextually relevant and accurate.

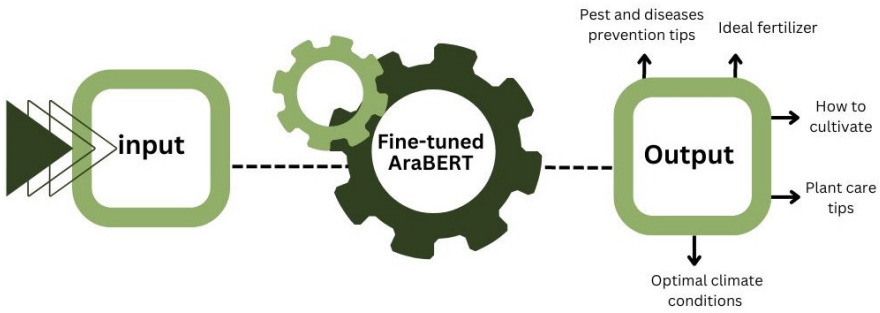


Figure 3.2: Mureq framework

This study will develop Mureq using BERT through several stages. Firstly, AraBERT, a pre trained Arabic BERT model will be utilized. This model is able to acquire a deep understanding of the Arabic language and the contextual relationships between words. Subsequently, the AraBERT model will undergo fine-tuning specifically on an Arabic question-answering dataset, tailored to the agricultural domain. Using a fine-tuned question-answering model, AraBERT will interpret queries, identify agricultural intents, and generate suitable responses.

Mureq that leverage the power of BERT, aims to provide comprehensive services to farmers, gardeners, and individuals interested in planting in Saudi Arabia. This chatbot, will offers

expert advice on how to cultivate and care for ornamental plants and trees, taking into account the optimal climate conditions for each species. Additionally, Mureq provides important tips for pests and diseases prevention that affect plants. It empowers users to make informed decisions, helping them optimize their gardening practices. In particular, Mureq offers insights on the ideal fertilizers to enhance soil fertility and ensure the healthy growth of plants.

3.4.1 Requirements

Training Mureq requires the following :

- An Arabic data collection that should include information about various agricultural topics, such as how to cultivate, pests and diseases prevention, ideal fertilizer and other relevant subjects specific to the agricultural domain in Saudi Arabia.
- hardware resources such as A powerful processor, a high-end Graphics Processing Unit (GPU) and sufficient Random Access Memory (RAM).

3.4.2 Data Collection

Mureq extensively utilizes a diverse dataset to function effectively. This comprehensive dataset is being collated from a variety of sources rather than a single one. A significant portion of the data it uses comes from the broader web, encompassing various websites, forums, and other digital platforms. In addition, Mureq also relies on data gathered from academic sources such as the College of Agriculture and Food at Qassim University and agricultural books. These deliberate choices allowed for the collection of relevant and domain-specific information required to develop a highly accurate and effective chatbot. The gathered information encompassed various aspects, including cultivation techniques, proper care, and optimal climate conditions for the successful cultivation of ornamental plants and trees. Moreover, the dataset included valuable insights and recommendations for preventing pests and diseases commonly affecting these plants, as well as guidance on the ideal fertilizers to promote healthy plant growth. This meticulous dataset collection process ensured a comprehensive representation of plant-related information, facilitating the training, validation, and testing of the chatbot model for accurate and reliable agricultural guidance.

3.4.3 Data Structure

When it comes to fine-tuning a model, the choice of data format or structure plays a crucial role in determining the effectiveness and efficiency of the process. The data format used for fine-tuning serves as the foundation upon which the model is trained to improve its performance on specific tasks. It must be carefully designed to capture the relevant information and ensure compatibility with the model architecture.

This section will provide a sample of the question-answering dataset used in this study to illustrate its structure and define each feature within it. By examining this sample, insights will be gained into the structure and composition of the complete dataset.

Figure 3.3 displays a single instance of the dataset in JSON format that consists of the following components:

- "title": This field represents the plant name.
- "category": This field provides the category of the plant, whether it is Indoor or Outdoor.
- "paragraphs": This field holds an array of paragraph objects.

Each paragraph object has the following fields:

- "context": This field holds detailed textual information about the plant.
- "qas": This field represents an array of question-answer pairs related to the given context.

Each question-answer pair (QA pair) object has the following fields:

- "question": This field holds a specific question related to the context.
- "id": This field provides a unique identifier for the question.
- "answers": This field is an array containing possible answers to the question.

Each answer object has the following fields:

- "text": This field contains the answer text.
- "answer_start": This field indicates the starting position of the answer text within the corresponding paragraph's context.

```
{
 "title": "نبات النحاس",
 "category": "داخلي",
 "paragraphs": [
  {
   "context": " تكون أوراق هذه الشجيرة كثيفة ذات لون نحاسي جذاب بينما ليس لأزهارها أهمية جمالية، وتنمو هذه الشجيرات على ارتفاع يصل إلى 2.5 متر وموطنها الأصلي هو الجزر الجنوبية لمنطقة الياسفك حيث تنمو بصورة كثيفة؛ رغم تحملها للملوحة العالية؛ حيث تتطلب رطوبة عالية ومحتوى عالياً من النيتروجين لتعطي نمؤا مشابهاً لنموها في المناطق الإستوائية. ولاتصمد النباتات إلا بعد أن تنتشر جذورها في التربة. وعند تعرض النبات للرياح وقلة الرطوبة فإن أوراقه تتساقط أو تحترق حوافها. وتتباين ألوان أوراق هذه الشجرة مابين اللون الفضي والأرجواني والأحمر وهو ما يجعلها تشد بصر الناظر إليها. وتعد الزراعة بواسطة العقل الخشبية هي الشائعة في استزراع هذه الشجيرات في فصل الربيع بوضع خليط من البتموس والبرليت في مكان يحافظ على مستوى الرطوبة.",
   "qas": [
    {
     "question": "ماهي طريقة زراعة نبات النحاس؟",
     "id": "11",
     "answers": [
      {
       "text": " تعد الزراعة بواسطة العقل الخشبية هي الشائعة في استزراع هذه الشجيرات في فصل الربيع بوضع خليط من البتموس والبرليت في مكان يحافظ على مستوى الرطوبة",
       "answer_start": 556
      }
     ]
    }
   ]
  }
 ]
}
```


Figure 3.3: Dataset instance

3.4.4 System Design Procedure

This section focuses on two main design components of the system namely, conversation flow and proposed methodology workflow. It provides an overview of how the system will handle the conversation for the overall process highlighting the key components and the order of operation.

3.4.4.1 Conversation Flow Design

This design component is responsible for organizing the chatbot's conversation flow. It is made up of a number of stages that are arranged in a certain order to ensure an effective conversation experience, as illustrated in Figure 3.4 and 3.5.

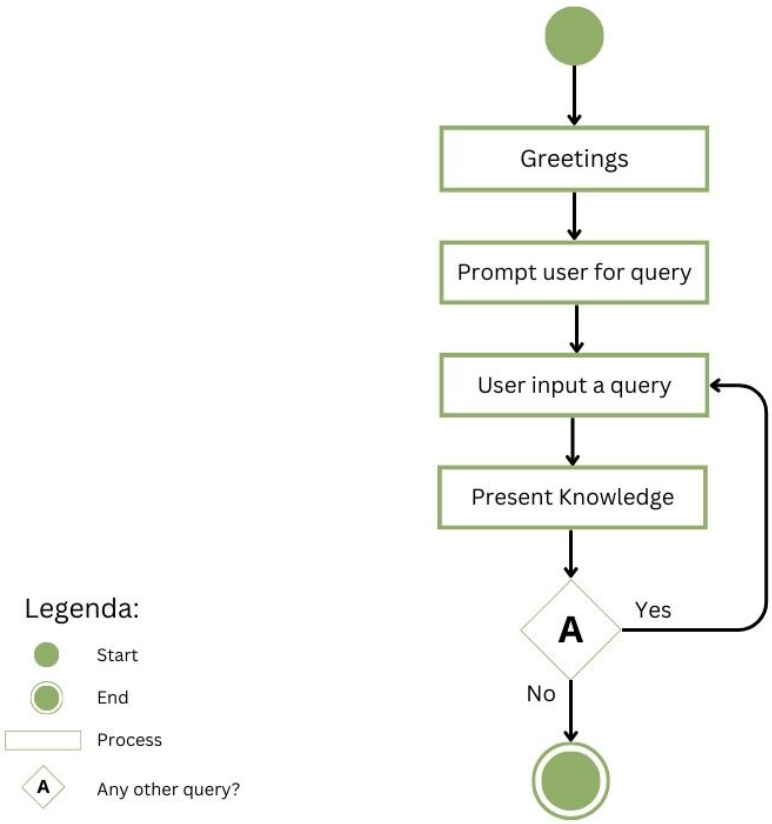


Figure 3.4: Flow of the conversation

Figure 3.5: Mureq interface that showcases the flow of the conversation

To begin with, the chatbot greets the user at the start of the conversation in order to establish a friendly tone and a positive rapport. The chatbot then prompts the user to provide his query. Once the user sends his query, the chatbot thoroughly examines the input to understand the user's question and extracts relevant information to address their needs accurately.

After that, the chatbot retrieves the relevant information from its knowledge base and presents it to the user in a coherent and understandable manner.

3.4.4.2 Proposed Methodology Design

This design component outlines the sequence of operations carried out by the AraBERT model which is the core of the chatbot, starting from receiving the input till presenting the response. As shown in Figure 3.6, the design component of the model consists of the following:

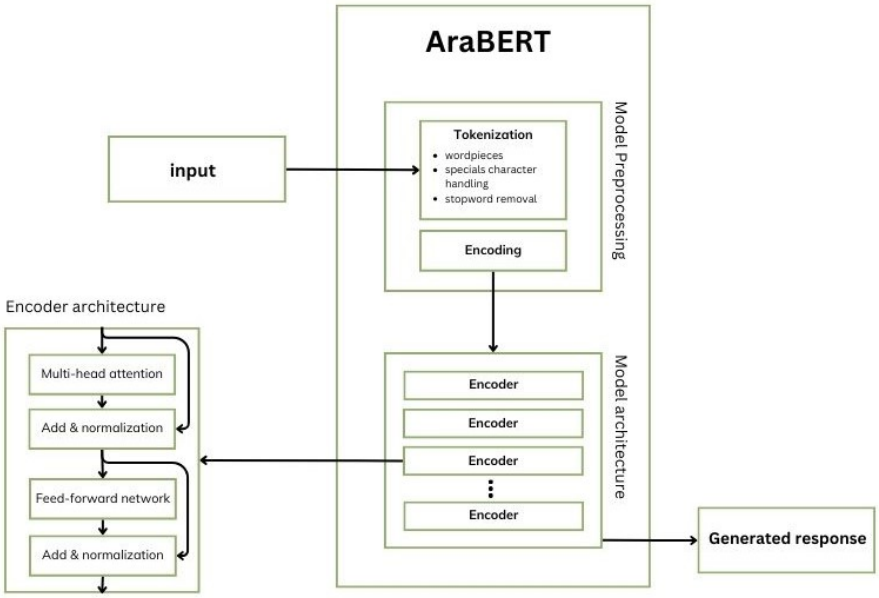


Figure 3.6: Proposed methodology design

To begin with, the system captures the input in the form of text, which is expressed using natural language. This process acts as the initial entry point for the system, paving the way for subsequent steps.

Once the input is captured, the AraBERT model will perform several preprocessing steps that are equipped within the model itself. To start off, AraBERT utilizes its own tokenizer

specifically designed for Arabic. Thistokenizer segmentsthe input text into smaller units called "wordpieces," capturing the structure and context of the Arabic language. It also handles special characters, punctuation and diacritics during tokenization step as well as stop words removal. Once the text is tokenized into wordpieces, AraBERT converts these wordpieces into numerical representations that can be processed by the model. Each wordpiece is assigned a unique index or embedding vector based on a predefined vocabulary.

After these representations are generated, they will go into a series of computational operations to understand and process them. This includes passing them through multiple layers of multi-head attention and Feed Forward Neural Networks(FFNNs) blocks, which enable the model to capture the contextual relationships between words. Then finally the model generates a response.

3.5 Summary

Overall, this chapter provides a detailed account of the study type, methodology approach, framework, data used and system design. It demonstrates a systematic and rigorous process undertaken to set the stage for the subsequent implementation and interpretation of the project.

Chapter 4

IMPLEMENTATION AND TESTING

IMPLEMENTATION AND TESTING

4.1 Introduction

This chapter discusses the implementation and testing of the Mureq chatbot. Covering various stages from the implementation process to system architecture and testing procedures. This chapter aims to provide insights into the methodologies and techniques employed in building a robust chatbot.

4.2 Implementation

This project utilizes the AraBERT model and fine-tunes it on a question-answering (QA) dataset focused on the agricultural domain. Several steps are undertaken to ensure optimal results from the model. The first step is data preparation, where a chain of steps are undertaken to align the dataset with the question answering task and ensure compatibility with the model. Once the dataset is ready, the next step is model training, where various strategies are applied to optimize the training process and improve model performance. Then model evaluation is carried on to assess the performance of the fine-tuned model where several evaluation metrics and methods are applied to gain insights into the model's strengths and weaknesses. Finally, the fine-tuned model is saved to enable future use and deployment, preserving its parameters and architecture for easy retrieval and utilization. In the subsequent sections, a more detailed discussion of each step will be provided.

4.2.1 Dataset preparing

Before the dataset is utilized for model training and evaluation, it undergoes a series of critical steps that significantly impact the model results. These steps are outlined as follows:

Step 1: Dataset splitting

After accumulating a substantial amount of data, it undergoes a process of division into three separate sets, each stored in its respective JavaScript Object Notation (JSON) file. This partitioning is achieved using the train_test_split method from the scikit-learn library, which randomly divides the dataset into training, validation, and testing sets according to specified ratios. This approach is essential for achieving reliable and accurate model assessments, ultimately contributing to the development of high-quality models.

Step 2: Data extraction

After splitting the dataset, it proceeds to a specially crafted function that extracts and then returns the contexts, questions, and answers form each dataset as a list as demonstrated in Figure 4.1. This is done to facilitate the following steps.

```
Training set informaiton:
Length of each training array is 847 elements
The first 5 Questions of Training array:
['ماذا يسبب تعرض النباتات لدرجات حرارة منخفضة؟', 'ماهي شجرة الليخ؟', 'ماهي طريقة زراعة شجرة الليخ؟', 'ماهي الأماكن المناسبة لزراعة شجرة الليخ؟', 'ماهو المناخ المثالي لزراعة شجرة الليخ؟']
**********
Validation set informaiton:
Length of each Validation array is 107 elements
The first 5 Questions of Validation array:
['ماهي نبتة الفيروكاكتوس ؟', 'ماهي طريقة زراعة نبتة الفيروكاكتوس ؟ ', 'ماهي طريقة العناية بنبتة الفيروكاكتوس ؟', 'ماهو المناخ المناسب لنبتة الفيروكاكتوس ؟', 'ماهي الأمراض التي تصيب نبتة الفيروكاكتوس؟']
**********
Testing set informaiton:
Length of each testing array is 108 elemants
The first 5 Questions of Testing array:
['ماهو نبات دراسينا؟', 'ماهي طريقة زراعة نبات دراسينا؟ ', 'ماهي طريقة العناية بنبات دراسينا؟', 'ماهو المناخ المناسب لنبات دراسينا؟', 'ماهي الأمراض التي تصيب نبات دراسينا؟']
```

Figure 4.1: A sample output of step 2.

Step 3: Add the 'answer_end' key after calculating its value

Another crucial step is the indication of the end index of the answer. This step is carried on by passing the 'context' and 'answers' lists that were extracted in the previous step to a helper function. This function recalculates the 'answer_start' key to ensure its correctness, then calculates the 'answer_end' and adds it to every answer dictionary in the 'answers' list as shown in the sample provided in Figure 4.2.

```
A sample of train answers array before adding 'answer_end' : {'text': 'ظهور بقع سوداء أو بيضاء غير منتظمة.', 'answer_start': 324}
A sample of train answers array after adding 'answer_end' :{'text': 'ظهور بقع سوداء أو بيضاء غير منتظمة.', 'answer_start': 324, 'answer_end': 359}
```

Figure 4.2: A sample output of step 3.

Step 4: Tokenization

A tokenizer is initialized using the Hugging Face transformers library, specifically through the AutoTokenizer class. The chosen model was AraBERT-based question-answering model which is specifically designed for the Arabic language and tailored for the question-answering task. Following this initialization, the tokenizer is employed to preprocess the training, validation, and test data. This preprocessing involves tokenizing both the contexts and questions. Additionally, it handles the truncation parameter, which ensures that sequences longer than the maximum length supported by the model are appropriately truncated. Also, the padding parameter ensures that all sequences are uniformly padded to the same length, typically with padding tokens resulting in producing encoded data. Moreover, the encoded data is enriched with the utilization of a helper function called 'add_token_positions'. This function calculates and then Integrates token positions for the start and end points of answers, offering essential information for training the question-answering model.

Step 5: Custom Data Class

A custom dataset class called 'QADataset' specifically designed for question-answering tasks is defined. The class inherits functionality from torch.utils.data.Dataset, a PyTorch class used for managing datasets. This class accepts encodings as input and can access each encoded item by its index using the '__getitem__' method. A sample of the output could be shown in Figure 4.3. After defining the class, an instance of each dataset's encoding was created.

```
An object sample of QAClass for train encodings:
 {'input_ids': tensor([    2, 13461, 34291,   200, 19673,  1585,   121, 12190, 28096,   103,
          1034,   306, 19304, 20950, 10564,  1281,   103, 58608,   181, 14227,
          2136, 28611,   103, 26667, 28611,   103, 35579,   305,  3631, 17197,
          1557,   305,  1282,   773,  1160,   103, 31801,   465,   233,   199,
         52117,   103, 17056, 54267, 10221,  1305, 47292,  3165,    20,  9733,
         47554, 26932, 31045,  1338,  2077,   420, 41768, 21136,   324,  9309,
           914, 57362,   315, 46055,   817, 10216,   201,  6912, 17243, 12368,
          1632,  6289,   383,  1377,   816, 56067,   198,  1693,   563,   829,
           819, 49675,   401,  1023,   802,    20,     3, 25830, 13461, 34291,
           200,   105,     3,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0]), 'token_type_ids': tensor([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0]), 'attention_mask': tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0]), 'start_positions': tensor(1), 'end_positions': tensor(47)}
```

Figure 4.3: A sample output of step 5.

Step 6: DataLoader

Lastly, the instances will be passed to a data loader, which facilitates shuffling and specifies the batch size for each dataset.

4.2.2 Model Training

The pre-trained Arabert model was loaded initially using 'AutoModelForQuestionAnswering. from_pretrained()'. It comprises an initial embedding layer followed by 12 encoder stacks. In order to expedite the training process and reduce the number of parameters that need to be learned, a decision was made to implement a freezing technique. Specifically, one of the encoder layers was frozen, meaning its weights and biases were not updated during the fine-tuning process. Applying this technique resulted in a reduction in the overall size of the model, which in turn facilitated faster saving of the model. To optimize the learning process, the AdamW optimizer was chosen and the batch size for all three datasets was set to be the same. Once the model hyperparameters had been appropriately set, the model was trained using a traditional training loop for predefined epochs. To prevent overfitting and promote efficient learning, an early stopping regularization technique was applied. This technique involved continuously monitoring the validation loss during training. If the validation loss failed to improve for three consecutive epochs, the early stopping mechanism was triggered. The purpose of early stopping is to prevent the model from overfitting the training data and to enable more efficient learning. After completing the training loop, the training and validation losses were plotted using matplotlib to gain insight of the results.

4.2.3 Model Evaluation

Following the model training phase, an evaluation process starts, wherein the model's efficacy in a question-answering task is measured. The evaluation loop calculates standard metrics like BLUE score, Exact Match (EM) score, and F1 score. The resultant scores furnish an overview of the model's proficiency in answering questions, facilitating informed assessments of its performance. Another method used to assess the model is by utilizing Pipelines from the Transformers library. This method offers a more user-friendly approach to testing the model's capabilities by letting users input a question along with a relevant context. The Pipeline then employs the model to process this input and returns the predicted answer to the user. This approach simplifies the testing process by providing a straightforward way to interact with the model and obtain answers.

4.2.4 Model Saving

As the concluding step, the model is saved using the save_pretrained() method offered by PyTorch and serves to serialize the model's parameters, architecture, and other essential details. Saving the model allows for retaining its state for future utilization, such as deployment or subsequent training, eliminating the need to retrain it entirely from scratch.

4.3 System Architecture

This section provides an overview of the system architecture. The query captured by the system undergoes a structured process, within a series of interconnected components that begins with text embedding and proceeding through retrieval and extraction stages, ultimately providing answers based on user input.

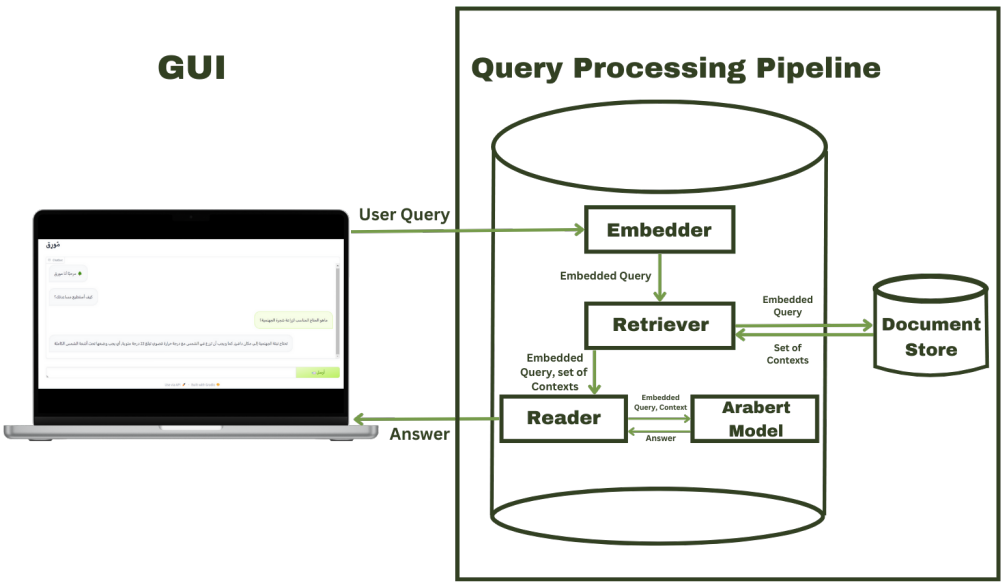


Figure 4.4: System Architecture Diagram

In the architecture diagram illustrated in Figure 4.4, the system comprises several key components facilitating the management of user queries and coordination of information flow. Upon user query submission through the interface, a sequence of operations is initiated by the pipeline. Initially, the query undergoes processing by the embedder, which utilizes a model that is re-

sponsible for converting the query into embeddings. These embeddings are then utilized by the retriever component to fetch relevant contexts from the document store. Subsequently, the retrieved contexts and embedded query are passed to the reader component. Then, the reader component extracts relevant context from the retrieved contexts and pass it with the embedded query to the fine-tuned AraBERT model. After that, this model extracts the answer span from the provided context. Finally, the reader component returns the answer to the user through the system's interface.

4.3.1 Components

- **Document Store:** A document store, also known as a document-oriented database, serves as a storage and retrieval system for managing semi-structured data, commonly referred to as documents [34]. One notable example of such a service that has been used in the system is Pinecone, which is utilized alongside Haystack to optimize document storage, retrieval, and querying processes. By leveraging Pinecone's advanced indexing and similarity search capabilities, this integration significantly enhances the speed and accuracy of document retrieval, particularly in large-scale applications. Additionally, Pinecone simplifies the provision of long-term memory for high-performance AI applications, offering a managed, cloud-native vector database with a user-friendly Application Programming Interface (API) and no infrastructure hassles. It efficiently serves fresh, filtered query results with low latency at the scale of billions of vectors [35].

- **Embedder:** A component responsible for converting textual data into numerical representations, often referred to as embeddings. The embedder in the system is responsible for processing user queries by converting them into embeddings using Sentence similarity model. These embeddings capture the semantic meaning of the queries, enabling effective retrieval of relevant documents by the retriever component.

- **Retriver:** It consists of fast and simple algorithms allowing identification of candidate passages or contexts from a large collection of documents [36]. Utilizing the embedding retriever within the system enables the retrieval of pertinent documents by assessing semantic similarity between user queries and document embeddings. The retriever is scalable and capable of handling extensive datasets efficiently. Additionally, it offers customization options such as specifying the number of retrieved documents (top_k) to meet specific system requirements.

- **Reader:** reader Analyzes retrieved contexts to pinpoint context containing relevant answers to user queries. By scanning through the contexts, it selects context based on their semantic alignment and contextual relevance.
- **Model:** The QA model integrated with Reader component. It scrutinizes selected context to identify answers aligned with the user's query. The model used in this system is a QA araBERT model fine-tuned for the agricultural domain.
- **Pipeline:** A sequence of interconnected components that facilitate the processing of user queries and retrieval of relevant information from documents. It comprises several key components, including an embedder, a retriever, and a reader. The pipeline orchestrates the flow of data through these components, starting with the processing of the user query by the embedder, followed by document retrieval using the retriever, and finally, the extraction of answers by the reader. This coordinated process ensures efficient information retrieval and accurate responses to user queries.
- **GUI:** Graphical User Interface (GUI) component, which facilitates user interaction with the system. It provides a user-friendly interface for inputting queries and receiving responses. Powered by the Gradio library, the GUI simplifies the development of interactive machine-learning applications, enhancing accessibility and usability for users.

These various components work together to establish a resilient system structure capable of efficiently managing user queries, retrieving pertinent information, and promptly presenting answers to the user.

4.3.2 Tools

This subsection delves into the libraries, packages, and frameworks used to develop this system, serving various purposes such as fine-tuning, evaluation, and visualization. Each component plays a crucial role in ensuring the system's success by streamlining tasks and enabling efficient execution of machine learning workflows while also addressing potential package conflicts.

4.3.2.1 Frameworksa used

- **PyTorch:** version 2.2.1+cu121 is chosen as the framework for fine-tuning the question-answering model due to its flexibility, ease of use, and rich features tailored for deep learning tasks.

- **Haystack:** version 2.2.1+cu121 is chosen as the framework for fine-tuning the question-answering model due to its flexibility, ease of use, and rich features tailored for deep learning tasks.

4.3.2.2 Libraries used:

- **Transformers:** is a powerful library for natural language processing tasks, offering access to a wide range of pre-trained models such as BERT, GPT, and more. Transformers version 4.38.2 has been used to introduce enhancements and bug fixes, ensuring improved performance and stability.

- **Matpltlib:** is a widely-used plotting library in Python for creating static, animated, and interactive visualizations. Matplotlib version 3.7.1 is applied for visualizing the training loss over epochs during the model training process.

- **Scikit-learn:** is a widely-used machine learning library in Python, offering tools for various tasks such as classification, regression, clustering, and more. Scikit-learn version 1.2.2 is applied for computing performance metrics like F1 score and accuracy during model evaluation.

- **tqdm:** is a Python library that provides a progress bar for iterables, allowing users to track the progress of loops or operations in real-time. tqdm version 4.66.2 is utilized to visualize the progress of training epochs during the model training process.

- **Sentence Transformers:** is a powerful Python library designed for computing embeddings of sentences and text passages. Sentence Transformers version 2.6.0 is employed to generate embeddings for textual data.

- **Gradio:** is a user-friendly Python library designed for building interactive web-based interfaces for machine learning models. Gradio version 4.23.0 is utilized to create a chat interface for the Arabic question-answering system.

4.3.2.3 Packages used

- **NumPy:** is a fundamental package for scientific computing with Python, providing support for multi-dimensional arrays and matrices, along with a collection of mathematical

functions to operate on these arrays. NumPy version 1.25.2 is applied for numerical operations and array manipulation tasks.

4.3.3 The Proposed Algorithm

This subsection presents the system's pseudocode, outlining its operational logic and functionality at a high level.

START

INITIALIZE document store with Pinecone API key

CONVERT data from JSON file to Haystack documents
  READ JSON data from file
  For each item in the JSON data:
    EXTRACT 'title' and 'paragraphs' information
    CREATE Haystack documents by combining 'title' and 'context' of each paragraph

INSERT documents into the document store
  EMBED documents using Sentence similarity model
  STORE documents with embeddings in the document store

INITIALIZE Pinecone Embedding Retriever using the previously initialized document store

INITIALIZE query pipeline
  INITIALIZE a pipeline for querying documents
  ADD components for text embedding and retrieval
  CONNECT components in the pipeline, passing embeddings from text_embedder to retriever

INITIALIZE an Extractive Reader with the 'Arabert' model

DEFINE response function for the chatbot interface
  PROCESS user messages using the query pipeline
  RETRIEVE relevant answers from the documents using the initialized reader
  RETURN responses to user messages

CREATE GUI Chat Interface using Gradio
  USE the defined response function to handle user queries and display responses in the interface

LAUNCH the chatbot interface

END

4.3.4 Testing

This section will present the metrics utilized to evaluate the performance of AraBERT-based question-answering (QA) model. These metrics provide valuable insights into the model's capabilities and accuracy in providing answers to questions. Through examination of these metrics, the effectiveness and overall performance of the QA model can be assessed.

Training Loss:

The training loss is a measure of how well a model is performing on the training data. It is calculated during the training process and is used to update the model's parameters to minimize this loss. In this study, the training loss score of 0.268 indicates that the model has achieved a relatively low level of loss on the training data.

Validation Loss:

Validation loss is a metric used to assess the performance of a model on a separate validation dataset during the training process. It measures the discrepancy between the model's prediction answers and the ground truth answers in the validation dataset. A validation loss score of 1.288 in this study indicates that the model has achieved a low level of loss on the validation dataset.

Exact Match:

Exact match is a score that gives a one if the predicted answer is identical to ground truth answer and a zero otherwise. That means EM is a binary score assuming values in 0, 1. This metric measures the proportion of questions where the model's predicted answer perfectly matches the ground truth answer, considering both start and end positions.

Formula:

$$EM = \frac{\text{(Number of questions with exact match predictions)}}{\text{(Total number of questions)}} [37]$$

In this study, the EM score 0.64, this indicates that approximately 64% of the questions in the evaluation dataset had the model's predicted answer perfectly aligned with the ground truth answer [37][38].

F1-score:

The F1-score aims to find the harmonic mean (balanced in importance) between recall and precision. It measures the word overlap between the predicted and the gold answer in a more flexible way than that used in EM, as it allows us also to trade off precision against the recall. However, it does not capture the similarity between two correct answers that differ in their semantic content. The F1-score is defined as follows:

$$F1 = \frac{2*(Precision*Recall)}{Precision+Recall)} [37]$$

This study results with 0.81 F1-score, this indicates that the model achieved a relatively high level of performance in terms of balancing precision and recall [37][38].

BLEU:

BLEU compares the n-grams of the answer with the n-grams of the reference by counting the number of matches. These matches are position-independent. Obtaining the BLEU score is a three step process. First, the logarithmic geometric mean of the modified n-gram precision,P_n , is calculated using n-grams up to length N and positive weights $w_n = \frac{1}{N}$ that summing to one

$$log(GM) = (\sum_{n=1}^{N} \frac{1}{N} logP_n) = log(\prod_{n=1}^{N} logP_n)^{\frac{1}{N}} \text{ [37]}$$

where $(\prod_{n=1}^{N} logP_n)^{\frac{1}{N}} = \prod_{n=1}^{N} log(P_n)^{w_n}$ is the geometric mean of the modified n-gram precision. Here the n-gram precision,P_n , is defined by

$$P_n = \frac{\sum_{c \in C} \sum_{n-gram \in c} Count_{clip}(n-gram)}{\sum_{c' \in C} \sum_{n-gram' \in c'} Count(n\ -\ gram')}$$

where c is the generated response (i.e., answer) that contains n-gram of text and appear in C, C is set of the generated output (i.e., responses), $Count_{clip}(n-gram) = max(n-gram \exists$ A) $\subset$ R, is the maximum number of times the given n-gram of A appears in any corresponding R references (i.e., gold answer), and $Count(n-gram)$is the total number of words in R and $\forall$ n-gram $\in$ c 0. Second, a brevity penalty is calculated by

$$BP = \begin{cases} 1, & if c > r \\ e^{(1-\frac{r}{c})}, & otherwise \end{cases}$$

where c is the length of the predicted answer and r is the effective reference answer length. Finally, the BLEU score is calculated by

$$\text{BLEU} = \text{BP} * \text{GM}$$

As default parameters the original BLUE used N = 4 and uniform weights $w_n = \frac{1}{N}$. In this study, the BLEU score 0.82, this indicates that the generated answers had a relatively high level of match with the reference answers when evaluating n-grams up to length N.

By collectively analyzing these metrics, a comprehensive understanding of the QA model's strengths and weaknesses is gained, enabling an evaluation of its overall performance and capabilities [37][39].

4.4 Summary

This chapter showcased the implementation and testing of the Mureq chatbot , which utilized pre-trained model and NLP techniques to process user queries and provide accurate answers.The system architecture highlighted the seamless flow of information within the chatbot. Furthermore, The evaluation metrics employed in assessing the model's performance provided valuable insights into its efficacy.

Chapter 5

RESULTS AND DISCUSSIONS

RESULTS AND DISCUSSIONS

5.1 Introduction

In this chapter, the results and discussions regarding the implementation and testing of Mureq chatbot are presented. It includes a thorough analysis of the model's performance and major findings from the study, along with a discussion related to the proposed work.

5.2 Analysis Result

This section investigates the fine-tuning process for question answering tasks using transformer-based models. The analysis compares the performance of multiple transformer-based models after fine-tuning various parameters. The objective is to identify the best model in terms of performance and get an insight into its configuration. In the conducted competition, two transformer-based models, namely AraBERT v2 and QA AraBERT, were subjected to fine-tuning experiments on the training set. The focus of these experiments was on fine-tuning three key parameters: batch size, learning rate and number of epochs. It is worth mentioning that the fine-tuning process also considered the utilization of early stopping and the manipulation of frozen layers as part of the hyperparameter selection. The selection of these hyperparameters aimed to minimize losses, address overfitting, and optimize performance towards reaching the local optima.

During the training process, the models underwent training for a total of 15 epochs. However, it was observed that satisfactory results could generally be achieved within the range of seven to ten epochs. This finding suggests that a relatively smaller number of epochs is adequate to obtain promising performance.

The outcomes of the experiments consistently indicated that the QA AraBERT model outperformed AraBERT v2. Notably, the F1 score, reaching 0.81, exhibited higher scores for the QA AraBERT model. Table5.1 provides a comparison of the results, which includes the models with their arguments and evaluation scores, while Figure5.1 illustrates the visual representation of the highest scores achieved by each model that can be observed.

Model	Bach Size	Learning Rate	Number of Epochs	Early Stopping	Number of Frozen Layers	F1 Score	EM Score	BLEU Score
AraBERT V0.2	16	0.00001	10	No	2	0.77	0.67	0.77
AraBERT V0.2	8	0.0001	15	Epoch 6	1	0.78	0.66	0.81
AraBERT V0.2	8	0.00001	15	Epoch 10	1	0.78	0.66	0.82
QA AraBERT	16	0.00001	15	Epoch 11	1	0.75	0.55	0.77
QA AraBERT	8	0.00001	15	Epoch 11	2	0.77	0.63	0.79
QA AraBERT	**16**	**0.0001**	**10**	**Epoch 7**	**1**	**0.81**	**0.64**	**0.82**

Table 5.1: Comparing the results of AraBERT v2 and QA AraBERT Model after Fine-Tuning Parameters

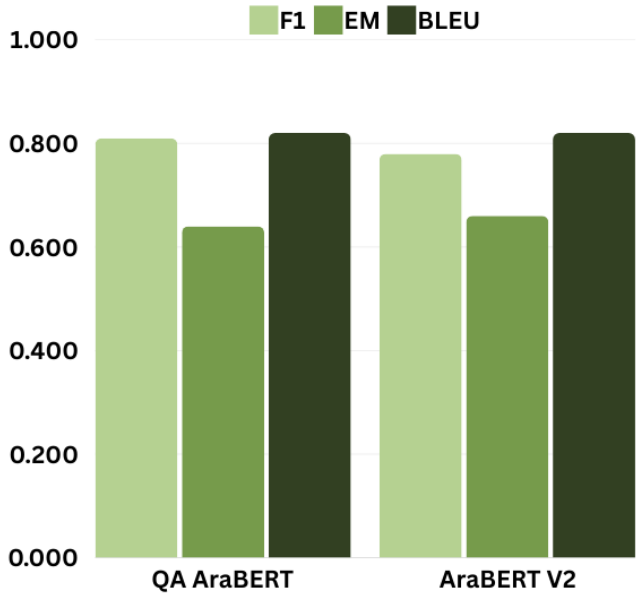


Figure 5.1: Representation of highest scores achieved by AraBERT v2 and QA AraBERT

Figure 5.2 shows the training process of the QA AraBERT model that achieved an F1 score of 0.81. It depicts a line graph of the train loss and validation loss, where is the x-axis representing the epoch and the y-axis representing the loss. As the number of epochs increases, both the training and validation loss generally decrease. The blue curve represents the average loss of the model on the training data after each epoch. The loss typically decreases as the model learns from the training data. In this figure, the train loss starts around 2.5 and decreases to around 0.2. This suggests that the model is effectively learning to perform the question answering task. The orange curve shows that the validation loss is decreasing over time. This indicates that initially, as the model is trained, its performance on the validation set improves. However, it is important to note that the rate at which the validation loss decreases is slower compared to the training loss (represented by the blue curve).

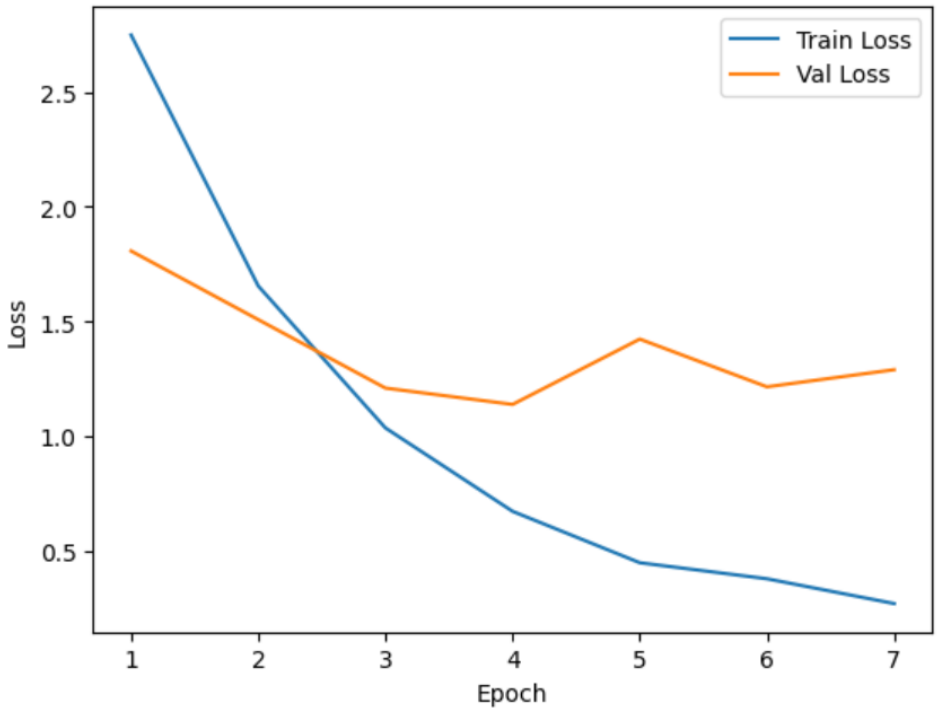


Figure 5.2: Training and validation loss curves for the QA AraBERT model that achieved 0.81 F1 score

In this training process, with a patience value of 3, early stopping was applied when the validation loss increased at epoch 4. The training process indeed stopped after epoch 7. This suggests that the model's validation loss kept increasing beyond the patience limit of 3, indicating overfitting and the need to stop the training process to prevent further degradation in performance.

5.3 Major Findings

This section presents the findings of this study, emphasizing AraBERT-based Question-Answering (QA) model refinement and a dataset in ornamental plants.

1. **Ornamental Plant Dataset:** A curated dataset of 128 ornamental plant species that grow in Saudi Arabia, categorized into indoor and outdoor plants. Figure5.3 visually represents the distribution of each category. In addition to containing the dataset some general information about plants, the dataset also includes specific information on 30 common pests and diseases that affect ornamental plants. Figure5.4, which is a visual representation of the dataset, shows the distribution of each sort of data. This comprehensive dataset offers researchers and plant enthusiasts a rich source of information

for studying and understanding the care requirements, pest management, and disease prevention of ornamental plants in Saudi Arabia.

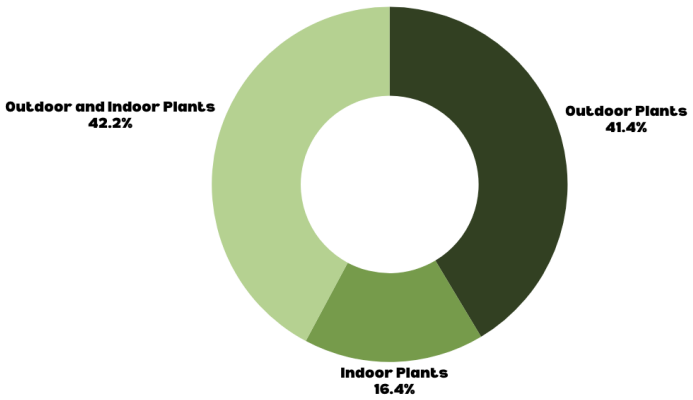


Figure 5.3: Distribution of Indoor and Outdoor Ornamental Plants in the dataset

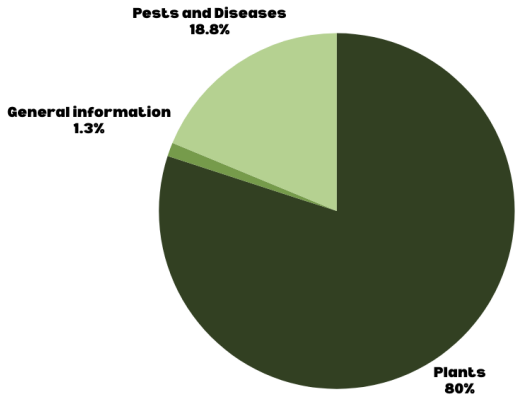


Figure 5.4: Distribution of data in the dataset according to its sort

2. **Enhanced ornamental plants Inquiry Resolution:** Fine-tuning the AraBERT-based Question-Answering (QA) model for the ornamental plant's domain resulted in a significant improvement in resolving ornamental plant queries. The Model exhibited F1-score of 81%. This enhancement underscores the efficacy of domain-specific training in catering to the nuanced terminology and context prevalent in ornamental plant inquiries, thereby empowering users with accurate and timely information to optimize ornamental plant practices.

5.4 Discussion Related to Proposed Work

This project introduces an Arabic agricultural chatbot, referred to as Mureq, where several fine-tuning techniques were employed to improve its performance. As identified previously, there was a lack of studies that implemented fine-tuning on the Arabic language model, however, three studies that implemented a similar approach were found for comparison. The comparison was based on two criteria: An Arabic Language model based on BERT and fine tuning on QA tasks.

The findings demonstrate that the presented model outperforms existing Arabic QA models in terms of F1 score and exact match, as shown in Figure 5.5 and Table 5.2. These results highlight the model's potential for accurate and reliable answers to agricultural questions in Arabic, with implications for knowledge dissemination and decision support for people in society.

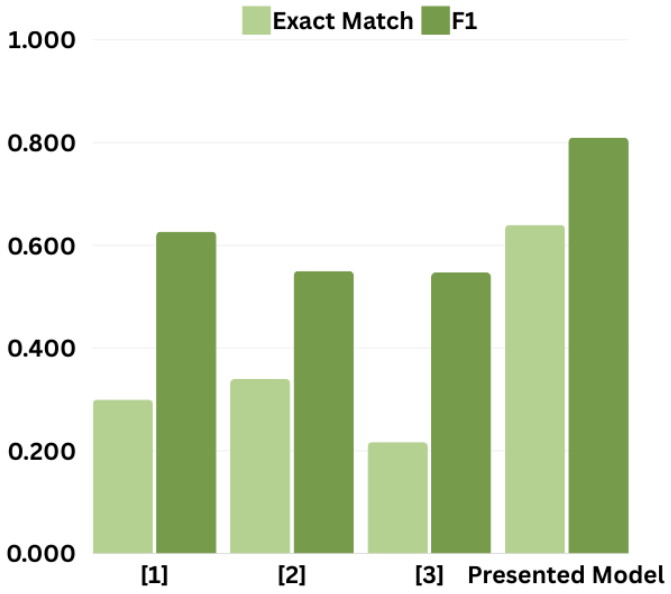


Figure 5.5: Visual representation for comparison of fine-tuned AraBERT Models introduced in different studies with the presented model

Paper	Model Version	EM	F1
[1][29]	AraBERT v1	0.30	0.627
[2][40]	AraBERT v2	0.34	0.55
[3][41]	AraBERT v2	0.217	0.547
Presented model	QA AraBERT	0.64	0.81

Table 5.2: Scores comparison of fine-tuned AraBERT Models introduced in different studies with the presented model

5.5 Summary

This chapter evaluates and analyzes the results of this study and presents its major findings. It demonstrates that fine-tuning a transformer-based model (QA AraBERT) led to superior performance in answering questions about ornamental plants. The chapter also presented a valuable dataset on these plants as a major finding for this study. Finally, the presented model surpassed existing Arabic question-answering models, demonstrating its potential as a reliable source of agricultural knowledge in Arabic.

Chapter 6

CONCLUSIONS AND FUTURE WORK

CONCLUSIONS AND FUTURE WORK

6.1 Introduction

In this concluding chapter, the study presents the outcomes of developing and evaluating the Mureq chatbot. It discusses its contributions, implications, limitations, and future research directions.

6.2 Conclusion

The project aimed to develop an agricultural chatbot named Mureq as its outcome, which provides knowledge and support in the field of agriculture, especially ornamental plants and trees in Saudi Arabia. This chatbot is powered by an AraBERT-based Question-Answering (QA) model, fine-tuned for the agricultural domain. This model was seamlessly integrated with a Retriever-Reader architecture, facilitated by the Haystack framework. The model exhibited commendable performance, particularly in the F1-score, which is considered the best metric for assessing the accuracy of the chatbot's responses, achieving a remarkable value of 0.81. This demonstrates the chatbot's ability to provide accurate answers. These results underscore the efficacy of the Mureq chatbot in efficiently retrieving and delivering accurate information within the agricultural domain.

6.2.1 Contributions and implications of the study

Developing an Arabic chatbot for ornamental plants significantly improves knowledge accessibility and promotes inclusive access to agricultural expertise among Arabic-speaking users. The chatbot offers guidance on plant care, cultivation techniques, and pest management, enhancing agricultural practices. Its availability in Arabic, given the language's complexity and limited agricultural resources in Saudi Arabia, effectively addresses linguistic challenges and fills information gaps in the field of ornamental plants. Notably, it is the first Arabic chatbot developed using AraBERT and integrated with the Retriever-Reader architecture in the field of agriculture, marking a significant advancement in ornamental plant practices. One of its benefits is that it spreads the culture of agriculture and increases green areas, which consequently improves the environment, quality of life, not only in Saudi Arabia, but in the Arab world as a whole, and for anyone who knows the Arabic language.

6.2.2 Limitations of the study

Despite the quality of the outcomes of this study, it faced limitations. This section outlines constraints encountered during development.

1. **Lack of Data**: Acquiring sufficient Arabic-language agricultural data proved challenging due to a lack of cooperation from agricultural government agencies. This challenge is widespread, as noted in the issues related to the current work, where limited, incomplete, or outdated data on Saudi Arabian plants hampers the chatbot's ability to be trained with accurate and up-to-date information.

2. **Hardware Constraints**: The fine-tuning process of the model demanded a significant GPU capacity, which may have limited the optimal performance achievable due to hardware constraints.

3. **Time Constraints**: The development process was hindered by time limitations, reducing the productivity of the model and constraining the amount of data that could be collected and utilized for training.

6.3 Future Work

In planning for the future development of the Mureq chatbot, several areas of improvement and expansion emerge. This section highlights key avenues for enhancing the chatbot's functionality and relevance within the Arabic-speaking community interested in ornamental plants.

- Fine-tune the model with specialized datasets and expand its knowledge base to cover a broader range in agricultural topics.

- Implement continuous monitoring and evaluation of the chatbot's performance and solicit feedback from users for iterative improvements.

- Incorporate the chatbot into a user-friendly platform or application aligned with initiatives like the Saudi Green Initiative, enabling seamless access and interaction for users across various devices and platforms.

6.4 Summary

In summary, this chapter has encapsulated the findings and implications of the study, shedding light on its significance within the agricultural field. Moreover, it has candidly addressed

the limitations encountered during the research process, providing valuable insights for future investigations.

Codes Reference:

References

[1] J. Bozic, O. A. Tazl, and F. Wotawa, "Chatbot testing using ai planning," in *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*. IEEE, 2019, pp. 37–44.

[2] "Saudi middle east green initiatives," https://www.greeninitiatives.gov.sa/ar-sa/sgi-forum/, 2023, accessed: 2023-11-24.

[3] H. Bansal and R. Khan, "A review paper on human computer interaction," *Int. J. Adv. Res. Comput. Sci. Softw. Eng*, vol. 8, no. 4, p. 53, 2018.

[4] A. Khanna, B. Pandey, K. Vashishta, K. Kalia, B. Pradeepkumar, and T. Das, "A study of today's ai through chatbots and rediscovery of machine intelligence," *International Journal of u-and e-Service, Science and Technology*, vol. 8, no. 7, pp. 277–284, 2015.

[5] A. Seza Doğruöz and G. Skantze, "How" open" are the conversations with open-domain chatbots? a proposal for speech event based evaluation," *arXiv e-prints*, pp. arXiv–2211, 2022.

[6] K. Nimavat and T. Champaneria, "Chatbots: An overview types, architecture, tools and future possibilities," *Int. J. Sci. Res. Dev*, vol. 5, no. 7, pp. 1019–1024, 2017.

[7] E. Adamopoulou and L. Moussiades, "An overview of chatbot technology," in *IFIP international conference on artificial intelligence applications and innovations*. Springer, 2020, pp. 373–383.

[8] "Architecture of artificial neural network - dot net tutorials," https://dotnettutorials.net/lesson/architecture-of-artificial-neural-network/, 2023, accessed: 2023-11-28.

[9] J. Weizenbaum, "Eliza—a computer program for the study of natural language communication between man and machine," *Communications of the ACM*, vol. 9, no. 1, pp. 36–45, 1966.

[10] K. M. Colby, *Artificial paranoia: A computer simulation of paranoid processes*. Elsevier, 2013, vol. 49.

[11] "jabberwacky - about thoughts - an artificial intelligence ai chatbot, chatterbot or chatterbox, learning ai, database, dynamic - models way humans learn - simulate natural human chat - interesting, humorous, entertaining." [Online]. Available: https://web.archive.org/web/20050405201714/http://jabberwacky.com/j2about

[12] R. S. Wallace, "The anatomy of alice in: Epstein, r., roberts, g., beber, g.(eds.) parsing the turing test," 2009.

[13] G. Molnár and Z. Szüts, "The role of chatbots in formal education," in *2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY)*. IEEE, 2018, pp. 000 197–000 202.

[14] M. B. Hoy, "Alexa, siri, cortana, and more: an introduction to voice assistants," *Medical reference services quarterly*, vol. 37, no. 1, pp. 81–88, 2018.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[16] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*.

[17] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

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

[19] R. Sukumar, N. Hemalatha, S. Sarin, and R. M. CA, "Text based smart answering system in agriculture using rnn," in *Proceedings of the 18th International Conference on Natural Language Processing (ICON)*, 2021, pp. 663–669.

[20] M. Dhyani and R. Kumar, "An intelligent chatbot using deep learning with bidirectional rnn and attention model," *Materials today: proceedings*, vol. 34, pp. 817–824, 2021.

[21] A. F. Hanifah and R. Kusumaningrum, "Non-factoid answer selection in indonesian science question answering system using long short-term memory (lstm)," *Procedia Computer Science*, vol. 179, pp. 736–746, 2021.

[22] Y. Alkhurayyif and A. R. W. Sait, "Developing an open domain arabic question answering system using a deep learning technique," *IEEE Access*, 2023.

[23] M. Z. U. Rehman, D. Raghuvanshi, and N. Kumar, "Kisanqrs: A deep learning-based automated query-response system for agricultural decision-making," *Computers and Electronics in Agriculture*, vol. 213, p. 108180, 2023.

[24] Y. W. Chandra and S. Suyanto, "Indonesian chatbot of university admission using a question answering system based on sequence-to-sequence model," *Procedia Computer Science*, vol. 157, pp. 367–374, 2019.

[25] W. X. Hui, N. Aneja, S. Aneja, and A. G. Naim, "Conversational chat system using attention mechanism for covid-19 inquiries," *International Journal of Intelligent Networks*, 2023.

[26] H. Honda and M. Hagiwara, "Question answering systems with deep learning-based symbolic processing," *IEEE Access*, vol. 7, pp. 152 368–152 378, 2019.

[27] I. V. Serban, C. Sankar, M. Germain, S. Zhang, Z. Lin, S. Subramanian, T. Kim, M. Pieper, S. Chandar, N. R. Ke *et al.*, "A deep reinforcement learning chatbot (short version)," *arXiv preprint arXiv:1801.06700*, 2018.

[28] H. Cuayáhuitl, D. Lee, S. Ryu, Y. Cho, S. Choi, S. Indurthi, S. Yu, H. Choi, I. Hwang, and J. Kim, "Ensemble-based deep reinforcement learning for chatbots," *Neurocomputing*, vol. 366, pp. 118–130, 2019.

[29] W. Antoun, F. Baly, and H. Hajj, "Arabert: Transformer-based model for arabic language understanding," *arXiv preprint arXiv:2003.00104*, 2020.

[30] T. Wael, A. Hesham, M. Youssef, O. Adel, H. Hesham, and M. S. Darweesh, "Intelligent arabic-based healthcare assistant," in *2021 3rd novel intelligent and leading emerging sciences conference (NILES)*. IEEE, 2021, pp. 216–221.

[31] S. Casola, I. Lauriola, and A. Lavelli, "Pre-trained transformers: an empirical comparison," *Machine Learning with Applications*, vol. 9, p. 100334, 2022.

[32] A. K. M. Masum, S. Abujar, S. Akter, N. J. Ria, and S. A. Hossain, "Transformer based bengali chatbot using general knowledge dataset," in *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2021, pp. 1235–1238.

[33] N. Bansal, A. K. Singh, A. Kumar, and A. Tyagi, "Comparative study of seq2seq and transformer model for chat bot," in *2023 IEEE 3rd International Conference on Software Engineering and Artificial Intelligence (SEAI)*. IEEE, 2023, pp. 125–131.

[34] C. Bell, *Introducing the MySQL 8 document store*. Springer, 2018.

[35] "Pinecone overview - pinecone." [Online]. Available: https://docs.pinecone.io/guides/getting-started/overview

[36] Z. H. Syed, A. Trabelsi, E. Helbert, V. Bailleau, and C. Muths, "Question answering chatbot for troubleshooting queries based on transfer learning," *Procedia Computer Science*, vol. 192, pp. 941–950, 2021.

[37] A. Farea, Z. Yang, K. Duong, N. Perera, and F. Emmert-Streib, "Evaluation of question answering systems: Complexity of judging a natural language," *arXiv preprint arXiv:2209.12617*, 2022.

[38] H. Schuff, "Explainable question answering beyond f1: metrics, models and human evaluation," Master's thesis, 2020.

[39] A. Chen, G. Stanovsky, S. Singh, and M. Gardner, "Evaluating question answering evaluation," in *Proceedings of the 2nd workshop on machine reading for question answering*, 2019, pp. 119–124.

[40] A. Alsaleh, S. Althabiti, I. Alshammari, S. Alnefaie, S. Alowaidi, A. Alsaqer, E. Atwell, A. Altahhan, and M. A. Alsalka, "Lk2022 at qur'an qa 2022: simple transformers model for finding answers to questions from qur'an," in *Proceedings of the OSACT 2022 Workshop*. ELRA European Language Resources Association, 2022, pp. 120–125.

[41] H. Alami, A. El Mahdaouy, A. Benlahbib, N. En-Nahnahi, I. Berrada, and S. E. A. Ouatik, "Daqas: Deep arabic question answering system based on duplicate question

detection and machine reading comprehension," *Journal of King Saud University - Computer and Information Sciences*, vol. 35, no. 8, p. 101709, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S131915782300263X