Global Scientific JOURNALS

GSJ: Volume 12, Issue 10, October 2024, Online: ISSN 2320-9186

www.globalscientificjournal.com

Intelligent Assistant Agents: Comparative Analysis of Chatbots through Diverse Methodologies

Menna Allah Reda
12422021688417@pg.cu.edu.eg

Abstract: Chatbots have rapidly evolved from rule-based systems to sophisticated AI models, transforming various industries and gaining more attention after advancements in Large Language Models (LLMs) and Language Models for Dialogue Applications (LaMDA). This is what made chatbots spread in many cultures and different languages such as English, Arabic, and other languages. This research paper reviews the history of chatbots, their efficiency, applications, various methods and components of their construction throughout history, and different examples of it, as well as Arabic and English chatbot applications, methods of measuring accuracy, and their various problems. It also compares the other research on chatbots in Arabic and English in Machine Learning and Deep Learning. It explores different algorithms employed in their substantial evolution, enhancing chatbots' intelligence, accuracy, and understanding of human speech and inquiries, ultimately aiding users more effectively. Also identifies key strengths, limitations, and emerging trends, aiming to inform stakeholders about chatbots' transformative potential and future directions in research and deployment.

Keywords: Chatbots, Intelligent Agents, Arabic Language, Artificial Intelligence (AI), Natural Language Processing (NLP), Large Language Models (LLMs), Machine Learning (ML), Deep Learning (DL)

Introduction

In the contemporary digital landscape, the proliferation of artificial intelligence (AI) technologies has caused the emergence of chatbots as intelligent agents capable of conversing with users in natural language. These chatbots, once rudimentary and rule-based, have evolved significantly, becoming indispensable personal assistants that aid individuals in various tasks, akin to virtual secretaries. Chatbots are utilized across various sectors to improve efficiency and customer experience and provide valuable assistance in different domains:

Companies, such as Delta and KLM use chatbots for handling customer inquiries, flight bookings, itinerary changes, and FAQs in the customer services domain. Buoy Health, provides medical advice, schedules appointments, and offers medication reminders in the healthcare domain. Educational institutions, like Georgia State University with its chatbot "Pounce," support students with course information, registration, financial aid inquiries, and campus resources in the Education domain. Banks and financial institutions leverage chatbots to offer personalized banking services, transaction alerts, and budgeting advice. Bank of America's virtual assistant, "Erica," assists customers with account management, bill payments, and financial planning in the finance domain. Additionally, chatbots are essential for improving the buying experience by suggesting products, order tracking, and customer support. Sephora's chatbot on Facebook Messenger helps users discover beauty products, receive makeup tutorials, and make purchases directly within the chat interface in the E-commerce domain. And too companies utilize chatbots for HR-related tasks such as employee onboarding, time-off requests, and benefits enrollment. Mya, an AI-powered recruitment assistant, assists job seekers with resume submissions, interview scheduling, and career advice in the Human Resources domain. Travel agencies and booking platforms too employ chatbots to assist travelers with trip planning, hotel reservations, and destination recommendations. Expedia's chatbot helps users find flights, hotels, and rental cars based on their preferences and budget in the travel domain. [31]

The concept of chatbots dates back to Alan Turing's Turing test in 1950, which spurred interest in artificial intelligence and human-computer interaction to assess a machine’s capacity to exhibit intelligent machine behavior similar to humans, the machine’s responses must be indistinguishable from those of a human during a five-minute test. Joseph Weizenbaum's ELIZA, the conversational agent in a very basic Rogerian psychotherapist, created in 1966, showcased the illusion of understanding through simple pattern matching.[31] Early chatbots like ELIZA and PARRY laid the groundwork for modern developments, with recent examples including A.L.I.C.E., Jabberwacky, and D.U.D.E., each offering unique functionalities and capabilities.[10][11] Chatbot development often falls within the realm of natural language processing (NLP), with some utilizing specialized languages or software like AIML. Some chatbots, like Jabberwacky, learn responses based on real-time interactions, while others optimize communication by combining evolutionary

algorithms with real-time learning. Chatbots may use also artificial neural networks, such as generative pre-trained transformers (GPT), which have become common for building sophisticated chatbots. Examples include ChatGPT and BioGPT. [31] Ever since Vaswani et al. first presented the transformer architecture in 2017, various language model-based transformers have emerged, including BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer) developed by OpenAI in 2018. These advancements in generative AI have significantly impacted our lifestyle, work dynamics, and interactions with technology. Recently, there has been substantial progress in the development of Large Language Models (LLMs) such as GPT-3, ChatGPT, GPT-4, and LaMDA (Language Model for Dialogue Applications). These models perform exceptionally well in tasks such as text summarization and question-answering, demonstrating remarkable accuracy.

Since late 2022, chatbot development has surged in popularity, driven by platforms like OpenAI's ChatGPT, Microsoft's Copilot, and Google's Gemini, which are based on large language models such as GPT-4 or Gemini, fine-tuned for specific tasks like simulating human conversation.[28] The landscape of chatbots continues to evolve, with developments such as Amazon's Q chatbot announced in November 2023, demonstrating ongoing innovation in the field. Speech-based chatbots like Siri, developed by Apple, provide users with voice-activated assistance and perform tasks based on spoken commands. These chatbots interpret and react to user inquiries in real time by utilizing speech recognition and natural language understanding technologies. [3]

ELIZA and PARRY were trailblazing conversational bots that operated on systems based on predefined rules and patterns. These systems proved useful for particular, targeted conversations but were not without their constraints. To overcome these, learning-based alternatives were created, divided into two main categories: retrieval-based and generative-based methods. Retrieval-based systems pick out answers from a pre-established database, whereas generative-based systems craft responses through deep learning.

Interaction through Menu-based utilizes a structured hierarchy for engaging users, accommodating different types of content such as images, links, and text. Learning-based strategies, like sequence-to-sequence (seq2seq) learning, utilize an encoder to reduce user input into a vector and a decoder to produce responses one word at a time. This technique has refined earlier methods and has been further developed with bidirectional recurrent neural networks (BRNNs) and attention mechanisms. The appearance of the transformer architecture, which incorporates an attention mechanism, has greatly enhanced the quality of responses and the efficiency of training by enabling more parallel processing. Transformer-based models, including GPT-3, GPT-4, BERT, and T5, have since reached the pinnacle of performance in various natural language processing tasks, signifying a significant breakthrough in the field of natural language understanding.[4]

In summary, chatbots have evolved significantly in recent years, offering diverse functionalities and applications across various domains, with ongoing advancements promising further innovation in the field. Despite this great development in chatbots and the technologies necessary to build them, it still faces many problems and obstacles, especially in Arabic chatbots. Progress is slow, and there are still many problems with it due to the nature of the Arabic language itself.

Methodology

This research paper undertakes a comparative analysis of chatbots by diverse methodologies.

Literature Review:

1. We will discuss in this research paper in the first section (1) the different types, techniques, components, and Challenges in chatbots at different times and we will review different papers to see the different elements.
2. In the second section (2) we will review Arabic and English chatbot papers to see different techniques, applications, limitations, and accuracy measurement ways. Also, get to know problems in Arabic chatbots and the characteristics of the Arabic language
1. that causes many problems in building the chatbots.
2. And in the third section (3) will review the intelligent chatbots and algorithms for building chatbots by Machine and Deep learning and their limitations.

Structured Prompt Search: A methodical process is employed to create customized prompts for each chatbot's features and uses. Detailed questions are developed to gather pertinent data from the internet, and scholarly articles.

Gathering Information: The data gathered from these prompt-based searches is methodically gathered and arranged. This involves collecting responses, overviews, examples, and other relevant information related to the set comparison standards.

Examining and Comparing: The gathered information is examined to assess the efficiency, design, and functionalities of chatbots in different scenarios. A comparative study is carried out based on the established criteria to point out the similarities, differences, and relative advantages.

Results: the paper analysis produces a comparative review of chatbots in different ways and scenarios in Arabic-English chatbots, intelligent chatbots, and in different applications. As a result of this comparison, focusing on the different problems in chatbots if due to language obstacles as in the Arabic language, and highlighting the most important problems in it or highlighting the various advanced chatbot problems and their strengths that help future researchers in developing chatbots to help users better.

1. Chatbots throughout the history

The chatbot is an application or a software program that is used to interact with humans and help them use natural language by simulating human conversation and one of the key goals of adopting chatbots is minimizing human involvement.[2][16] The conversation can be engaging at times with large vocabularies and a broad range of conversational topics.[22] Conversational agents, commonly known as chatbots, have become integral to our daily lives. They serve as personal assistants because they are like assistants

to humans in many tasks and fields.[3] The growing reliance on intelligent agents, such as chatbots, in daily life makes their integration into social media and instant messaging ecosystems increasingly commonplace and rational.[4]
Throughout history, text and voice chatbots have differed greatly in many respects, from the first that the non-smart chatbot used a rule-based way until it became an intelligent chatbot trying to understand the speaker's words to give him the appropriate smart answer to the context of his speech as a human does, and therefore chatbots went through many changes to that during that time. Therefore there are many definitions of chatbot types, components, Techniques, and challenges that have appeared with different researchers and we will talk about some of them in this paper.
These components and techniques collectively enable chatbots to understand user inputs and generate appropriate responses, thereby enhancing user experience and interaction quality. And also the limitations to face it for better able to understand chatbots. Addressing these limitations requires ongoing research and development efforts to enhance the capabilities and robustness of chatbot systems. By overcoming these challenges, chatbots can fulfill their potential as valuable tools for automating tasks, enhancing user experiences, and facilitating seamless interactions across various domains and industries.

1.1 Types of Chatbots:

There are various types of chatbots and differ from one paper to another based on the Functional of the chatbot, so we can categorize chatbots into two types based on various functionalities, design approaches, and purposes:

Based on Functional and Design Approach Categories:

Chatbots are typically organized into different components or modules, each serving a specific role to ensure the chatbot operates efficiently and accurately.

1. **Responder:**
   - Facilitates communication between users and the chatbot, managing input and output.
   - **Example:** If a user asks a question, the Responder interprets the query and decides how to respond, whether by fetching information, asking for clarification, or providing feedback.
2. **Classifier:**
   - Filters, segments, and normalizes user input before further processing.
   - **Example:** If a user types "What's the weather like?" or "Tell me about the weather," the Classifier recognizes that both queries are asking about the weather and normalizes the input for consistent processing.
3. **Graphmaster:**
   - Handles pattern matching and organizes the chatbot's knowledge base.
   - **Example:** If the chatbot has been programmed with specific responses to various patterns, such as "Hello," "Hi," or "Good morning," the Graphmaster would identify the greeting pattern in the user's input and trigger the corresponding response.[12]

Based on Specific Functionalities:

1. **Rule-Based Chatbots (closed domain):**
   They are programmed to respond to predetermined, targeted inquiries at the outset. In this type of chatbot, there aren't many input possibilities available to users. That's why it's closed domain.
   - Operate on predefined rules and patterns.
   - Look for particular keywords or phrases in user input.
   - Provide predetermined responses based on rules.
2. **AI-Powered Chatbots:**
   Leverage artificial intelligence techniques like machine learning and deep learning, dynamically understand and respond to user queries and continuously learn from interactions to improve responses.
   - **Deep Learning Chatbots:** Use neural networks to process and generate responses.
   - **End-to-End Systems:** Handle the entire conversation process without predefined rules or handcrafted responses. Instead, these systems use advanced machine learning techniques, such as deep learning, to understand and generate responses.
   - **Sequence-to-Sequence Models:** Translate input sequences into output sequences and are suitable for tasks like language translation and dialogue generation.
3. **Context-Aware Chatbots:**
   - Consider contextual information such as user location and time of day.
   - Tailor responses based on previous interactions to provide more personalized and relevant assistance.
4. **Transactional Chatbots:**
   - Facilitate transactions within conversations.
   - Allow actions like purchasing products, making reservations, etc., through a chat interface.
5. **Informational Chatbots:**
   - Act as virtual assistants or knowledge bases.
   - Provide information or answer queries on various topics leveraging vast repositories of data.[13][14]
6. **Corpus-based Chatbots:**

- **Retrieval-based approach (closed domain):** Uses information retrieval and heuristics to find the best response from a predefined set based on context and keywords. That's why it's closed domain.
- **Generation-based approach (open domain):** Generates new responses based on dialogue context, considering current and previous interactions. This approach doesn't need predefined responses but requires a large training set and has a higher risk of errors. That's why it's an open domain.

7. **Hybrid Chatbots** (open domain)**:**
   - Combines generation-based and retrieval-based methods to enhance response accuracy and diversity. Researchers rank responses from both models or improve retrieval responses with generative techniques for more informative interactions. And that's why it's also an open domain. [2][3]
   - The hybrid chatbot combines by more than one method together; it can combine by any methods to improve the chatbot accuracy.

1.2 Components of Chatbots:

Chatbots typically consist of several key components that work together to enable communication with users, generate appropriate responses, and engage in meaningful conversations, while continually improving their capabilities. These components may vary depending on the complexity and functionality of the chatbot, but here are some common components:

1. **Natural Language Processing (NLP):**
   - This component enables chatbots to understand and generate human-like language. It includes subtasks such as natural language understanding (NLU) to interpret user input and natural language generation (NLG) to produce coherent responses. And it's from the most used components and very important in chatbots these days.
   - **Natural Language Understanding (NLU):** NLU is the component responsible for processing user input and understanding the meaning and intent behind it. It involves techniques such as text parsing, entity recognition, and sentiment analysis to extract relevant information from user messages.
   - **Natural Language Generation (NLG):** NLG is the component responsible for generating responses to user inputs. It takes the output of the dialogue management system and converts it into natural language text that can be understood by the user. NLG may involve generating responses from predefined templates, using machine learning models to generate text, or combining both approaches.[13][15]
2. **Pattern Recognition:**
   - Chatbots use pattern recognition techniques to analyze user input and identify patterns or keywords that trigger specific responses. This helps them understand user queries and generate appropriate replies.
3. **Semantic Web:**
   - Leveraging semantic technologies, chatbots can understand the meaning of information and express relationships between concepts in a machine-readable format. This enhances their ability to interpret user queries and provide contextually relevant responses.
4. **Data Mining:**
   - Chatbots employ data mining techniques to analyze large datasets and extract valuable insights. By mining relevant information from vast repositories of data, they can improve their understanding of user preferences and behavior, leading to more personalized interactions.
5. **Text-Aware Computing:**
   - This component involves incorporating contextual information from the user's environment to provide more personalized and relevant responses. By understanding the context of the conversation, chatbots can tailor their responses to better meet the user's needs and preferences.[13]
6. **Front-end:**
   - This component interacts with users, utilizing NLP and AI algorithms to understand user intent and generate appropriate responses.
7. **Back-end:**
   - Responsible for managing the knowledge base, which contains domain-specific information necessary for generating responses.[14]
8. **Knowledgebase:**
   - Stores domain knowledge in a structured format accessible to the front end. The knowledge base is a repository of information that the chatbot can access to provide answers to user queries. It may contain structured data, or unstructured data. [14][15]
9. **Corpus:**
   - Represents the dataset or domain-specific data used for training and improving the chatbot's understanding and response generation capabilities.
10. **Dialog Management:**
    - Dialogue management handles the flow of conversation between the chatbot and the user. It determines how the chatbot responds to user inputs based on the current context of the conversation and any previous interactions.

Dialogue management may involve maintaining a conversation state, handling interruptions, and managing fallback responses.

11. **Sentiment Analysis:**
    - Understands the emotional context of user inputs to provide more empathetic responses.
12. **Integration with APIs:**
    - Accesses external services or data sources to enhance chatbot functionalities.
13. **User Interface (UI):**
    - The user interface is the interface through which users interact with the chatbot. It can be a text-based interface, such as a messaging platform or a chat window on a website, or it can be voice-based, allowing users to interact with the chatbot through speech.
14. **Integration with External Systems:**
    - Integrates with external systems or APIs to access additional information or perform tasks on behalf of the user. This may include accessing databases, retrieving data from third-party services, or triggering actions in other systems.
15. **Analytics and Monitoring:**
    - Analytics and monitoring components track the performance of the chatbot, including metrics such as user engagement, conversation completion rates, and user satisfaction. This information can be used to evaluate the effectiveness of the chatbot and identify areas for improvement.[14]

Choosing the best components for our chatbot from that component depends on the chatbot uses we do not use all components in all chatbots, it depends on us and which components we need and want to combine.

1.3 Chatbot Techniques:

These techniques collectively enable chatbots to understand user inputs, generate appropriate responses, and engage in meaningful conversations, thereby enhancing user experience and interaction quality. And there are very unique techniques that have appeared through the time, from those techniques:

1. **Parsing:**
    - Converting a text to less complicated terms
    - Analyzing and manipulating input text using NLP functions.
    - Example: Tree structures in NLTK.[12]
2. **AIML:**
    - Utilizes a markup language for conversational modeling.
    - Represents the knowledge as objects derived from XML
3. **Chat Script:**
    - Provides syntax for building sensible default responses.
4. **SQL and Relational Databases:**
    - Used for storing and retrieving conversation history, enhancing continuity in dialogue.
5. **Markov Chain:**
    - Generates probabilistic responses based on textual datasets.
6. **Language Tricks:**
    - Incorporating phrases or responses to add variety and enhance the Chatbot's conversational skills.
7. **Ontologies:**
    - Represents interconnected concepts for reasoning in natural language.[12]
8. **Neural Machine Translation (NMT):**
    - Enhances translation and alignment performance using techniques like Bidirectional Recurrent Neural Networks (BiRNN).
9. **DialogFlow:**
    - Uses AI-powered natural language understanding to process user inputs and generate responses.
10. **Neural Reinforcement Learning (RL):**
    - Enables the chatbot to learn and improve over time by rewarding desirable conversational properties. Optimizes conversational strategies based on user feedback.
11. **Firebase Real-time Database:**
    - Provides a scalable and reliable backend solution for storing chatbot data and user details.
12. **Android Studio:**
    - Used for developing mobile applications integrating DialogFlow and Firebase for Android devices.[14]
13. **Rule-based Techniques:**

Rely on predefined rules or patterns for responses.

- **Pattern Matching**: This technique involves creating rules or patterns that match specific user inputs. These patterns can be simple keywords, phrases, or more complex regular expressions. When a user input matches a pattern, the corresponding response associated with that pattern is triggered with predefined structures of responses.
- **Template-Based Responses**: Responses are generated using predefined templates that include placeholders for variables. These variables can be filled in dynamically based on user inputs or contextual information.
- **Eliza-Style Pattern Matching**: Inspired by the early chatbot Eliza, this approach identifies keywords or phrases in user inputs and generates pre-programmed responses based on predefined patterns.
- **Scripted Conversations**: Chatbots follow predetermined conversation flows and responses scripted by developers. These scripts dictate the progression of the conversation and the appropriate responses for different scenarios.
- **Keyword-Based Responses**: Responses are triggered by specific keywords or phrases detected in user inputs. Chatbots scan user messages for these keywords and provide predefined responses associated with them.
- **Decision Trees**: Conversation flows and decision-making processes are structured using tree-like structures. Chatbots navigate through different branches of the decision tree based on user inputs and system prompts.
- **Handcrafted Rules and Heuristics**: Developers manually define rules and heuristics based on domain knowledge to guide chatbot behavior. These rules dictate how the chatbot responds to specific user inputs or situations.
- **Finite State Machines**: Chatbot behavior and conversation states are modeled using finite state machines (FSMs). Each state represents a distinct stage in the conversation, and transitions between states occur based on user inputs and system triggers.[15]

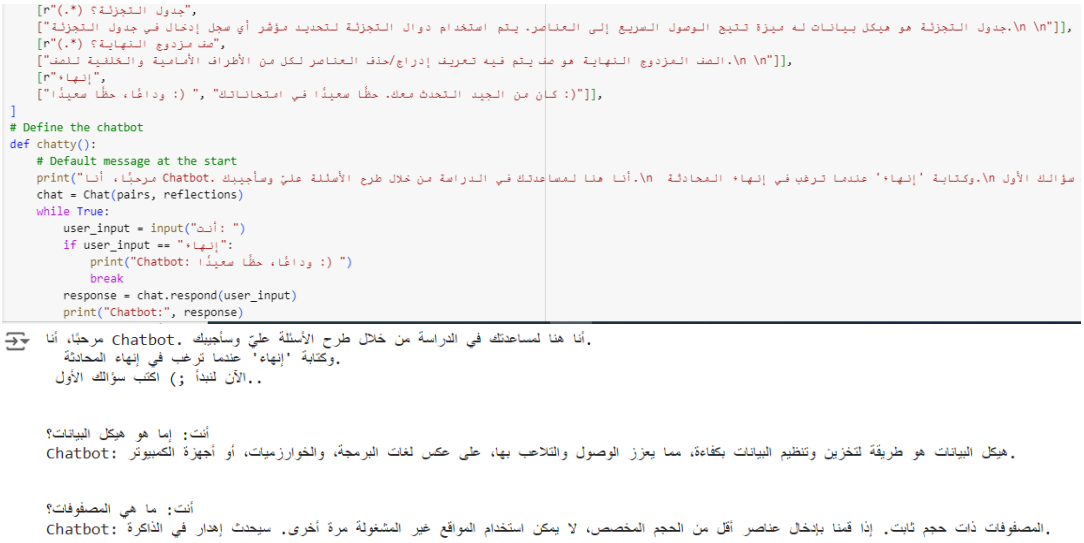

Fig 1.Chatbot with Rule-Based responses

14. **Artificial Intelligence (AI) Techniques:**
    - **Natural Language Processing (NLP):** This technique can analyze and understand human language inputs.
    - **Machine Learning Models:** Chatbots leverage machine learning algorithms to learn from data and improve their performance over time. This includes supervised learning, where chatbots are trained on labeled datasets of user interactions, and unsupervised learning, where they identify patterns and insights from unstructured data.
    - **Deep Learning Models:** Uses Deep learning models like RNNs and transformers for natural language tasks.
    - **Pretrained Models:** Advanced machine learning models that have been trained on large datasets, have very high performance, are capable of handling a wide range of tasks, and can be fine-tuned for specific tasks, like GPT-2, DIALOGPT. [2][15]
15. **Hybrid Approaches:**
    - Combines rule-based techniques with machine learning methods to leverage the strengths of both approaches. Rule-based systems provide structure and reliability, while machine learning models offer adaptability and context awareness. And can combine any techniques.[2][15]
16. **Intent Recognition:**
    - Techniques such as keyword analysis, semantic parsing, and probabilistic models are used to categorize user queries into predefined intents or actions. This allows chatbots to understand the underlying intention or purpose behind user inputs and provide appropriate responses.
17. **Entity Extraction:**

- Entity extraction involves identifying specific entities or objects mentioned in user inputs, such as names, dates, locations, or products. Named entity recognition (NER) and part-of-speech tagging are common techniques used for entity extraction.

18. **Context Management:**
    - Context management involves maintaining and updating the context of a conversation to ensure continuity and relevance. Chatbots keep track of previous interactions, user preferences, and ongoing conversation threads to provide coherent and meaningful responses.
19. **Sentiment Analysis:**
    - Sentiment analysis aims to identify the emotional tone or sentiment expressed in user inputs. Machine learning classifiers and deep learning models are used to classify text inputs as positive, negative, or neutral.
20. **Multi-turn Dialogue Handling:**
    - Multi-turn dialogue handling enables chatbots to engage in multi-step conversations with users. Techniques such as state tracking, dialogue policy learning, and response generation are used to manage complex dialogues and maintain coherence across multiple interactions.
21. **Knowledge Graphs:**
    - Knowledge graphs represent structured knowledge about a domain or topic in the form of interconnected entities and relationships. Chatbots use knowledge graphs to organize and access domain-specific knowledge, enhancing their understanding and reasoning capabilities.
22. **Emotion Generation:**
    - Emotion generation involves imbuing chatbot responses with emotional cues or expressions to enhance user engagement and empathy.[15]

1.4 Challenges and Limitations of Chatbots

The development and deployment of chatbots represent an exciting frontier in technology, closely tied to advancements in artificial intelligence and machine learning. While chatbots offer numerous benefits, they also face several limitations that must be considered for their effective use across various industries and applications. These limitations can be categorized into several key areas, like in Table 1:

Table 1.Limitations of Chatbots

Technical and Functional Challenges	1. **Complexity** o Requires significant programming expertise and a large knowledge base.[12] 2. **Knowledge Base Quality** o Effectiveness heavily depends on the quality and breadth of the knowledge base.[12] 3. **Context Understanding** o Struggles with comprehending complex contexts or nuanced language, leading to inaccurate responses.[12][13][14] 4. **Understanding Complex Queries** o Difficulty in interpreting complex or ambiguous user queries.[13] 5. **Integration with Other Systems** o Technical challenges in integrating with existing systems and databases, affecting functionality.[13] 6. **Vocabulary Limitations** o Struggles with handling unknown or rare words, leading to generic or inconsistent responses.[14] 7. **Short-sightedness** o Produces responses that may ignore future conversational directionality.[14] 8. **Fixed Input/Output Database** o Reliance on pre-programmed databases, limiting accuracy and relevance. 9. **Language Processing Challenges** o Struggles with accents, slang, and grammatical errors. 10. **Limited Conversational Abilities** o Operates within predefined flows, struggling with dynamic or open-ended conversations. 11. **Difficulty with Non-linear Interactions** o Struggles with non-linear or complex conversations.
Development and Resource Challenges	12. **Training Data Requirements** o Access to large volumes of training data is essential but challenging to obtain. 13. **Resource Dependency** o Requires extensive human-annotated resources, limiting flexibility and adaptability.[14] 14. **Limited Domain Knowledge** o Typically designed for specific domains, lacking breadth for diverse topics.[15] 15. **Dependency on Training Data** o Requires large amounts of training data, with performance suffering in unfamiliar

	scenarios.[15] 16. **Data Efficiency and Time** o Limited or no datasets in certain domains hinder chatbot development, especially for task-oriented systems. For example, the lack of Arabic resources slows the progress of Arabic chatbots. [2]
User Experience and Ethical Challenges	17. **User Experience** o Need for intuitive, responsive, and user-friendly interfaces to engage users effectively. 18. **User Comfort and Trust** o Some users may distrust chatbots, perceiving them as impersonal or lacking human expertise. 19. **Difficulty Handling Emotional Nuances** o Challenges in understanding and responding to emotional or nuanced inputs.[15] 20. **Lack of Emotions** o Predefined conversations limit chatbots' linguistic intelligence, making them mechanical. Incorporating emotions can enhance human-like behavior and appropriate responses.[2] 21. **Ethical Considerations** o Raises concerns related to privacy, data security, and bias.[15]
Evaluation and Performance Challenges	22. **Turing Test Limitations** o No chatbot has achieved a gold medal in Turing Test competitions like the Loebner Prize. 23. **Evaluation Challenges** o Metrics may not capture desirable conversational properties accurately.[14] 24. **Ambiguity in Conversation** o Natural language's multiple interpretations make simulating human conversation challenging. Current deep learning advancements still fall short of achieving human-like conversations, especially with ambiguous word senses. [2] 25. **Consistency in Interpretation** o Inconsistent interpretations lead to inaccurate responses. For example, the Microsoft Tay chatbot incident highlights this issue, where it was shut down after 16 hours due to misinterpretations.[2] 26. **Maintaining Conversation Flow** o Chatbots struggle with open-domain conversations, determining topics, and recognizing user intent, leading to user frustration.[2]
Security and Privacy Challenges	27. **Privacy and Security** o Chatbots relying on APIs must secure user data, posing privacy concerns. Ethical issues arise from misuse by either users or chatbots storing user information. [2]
Open Research Problems	27. **Datasets** o Word sense disambiguation increases computational complexity, and most research builds proprietary datasets, especially for Arabic. More annotated datasets are needed.[2] 28. **Evaluation Framework** o A comprehensive evaluation framework is necessary, considering multiple semantic meanings and conversation lengths across different domains. The absence of a universal evaluation framework makes testing and comparing chatbots difficult. Human evaluations are subjective and time-consuming.[2] 29. **Human-like Conversation** o Current research struggles with natural conversation generation. Factors like ambiguous inputs, multilingual capabilities, and leveraging emotional and contextual cues need further investigation.[2] 30. **Extended Perspectives** o The focus on specific research leaves other perspectives unaddressed. Including articles from more databases could provide additional insights.[2] 31. **Empirical Investigation** o Future works should involve empirical contributions to provide a broader analysis of chatbot development and usage.[2]
Challenges in selecting and implementing algorithms for chatbot development	32. **Algorithm Effectiveness**: o The effectiveness of each algorithm may depend on factors such as the complexity of the dataset, the nature of user queries, and the specific requirements of the application. [21] 33. **Trade-offs in Algorithm Selection**: o The choice of algorithm may involve trade-offs between accuracy, computational efficiency, and scalability. This requires careful consideration of the specific needs and constraints of the chatbot application.[21]

	34. **Training Data Quality**: o Limitations may also arise from the quality and quantity of the training data, as well as the preprocessing techniques applied to the data. Poor-quality data or inadequate preprocessing can significantly impact the chatbot's performance.[21]

1.5 Problems and Advantages in Specific Chatbot Techniques

Each technique presents unique advantages and challenges. Rule-based systems are simple but rigid, while advanced AI methods like LSTM and seq2seq offer flexibility but demand high computational resources. The choice of technique should align with the specific needs and constraints of the application, underscoring the importance of ongoing research and development to overcome these limitations.[2]

Table 2.Advantages and Disadvantages of Chatbots techniques

Chatbot building Techniques	Advantages	Disadvantages
Rule-Based	- Simple and lower-cost implementation. - Quick deployment. - No overtime for learning user intent.[2]	- **Inability to Learn or Adapt**: Cannot learn on their own. Rule-based chatbots cannot improve over time as they don't learn from interactions. Fail to respond outside its preset understanding. - **Limited to Predefined Rules**: They operate strictly within the confines of the rules set by the developers, making them unable to handle inputs that fall outside of these predefined rules.[2]
Pattern matching	- Sufficient on simple tasks. - Select informative responses from candidate responses. - More flexible than the rule-based.[2]	- Providing chatbots without reasoning and creation. - Limited capabilities and repeated responses.[2]
Parsing	- Providing dependency relationship between words or semantic structure of the text.[2]	- **Same Issues as Rule-Based and Pattern Matching**: Shares the fundamental limitations of rule-based systems. - **Complexity in Handling Dependency Relationships**: Parsing involves breaking down sentences into more manageable components, which can add complexity without improving the chatbot's ability to understand or generate nuanced responses.[2]
Retrieval-Based	- Flexibility over rule-based systems.	- **Lacks Reasoning and Creativity**: These chatbots cannot generate new responses or understand context beyond predefined patterns. - **Repetitive and Limited Responses**: They often produce the same responses to similar inputs, which can make interactions monotonous and less engaging for users.[2]
AIML (Artificial Intelligence Markup Language)	- Advantages of pattern matching. - Powerful in designing conversational flow.[2]	- **Manual Creation of Patterns**: Requires extensive manual effort to define all possible conversation patterns. - **Difficult to Scale**: As the number of possible interactions increases, scaling the AIML approach becomes impractical due to the need to manually account for each potential conversation.[2]
Generation-Based (LSTM, CNN, etc.)	- Accurate on a larger dataset. - Suitable to remember longer sequences.[2]	- **High Resource Consumption**: These models require significant computational resources for both training and execution.[2] - **Long Execution Times**: More complex structures and Complex architectures like LSTMs and CNNs take longer to process inputs and generate responses. And Uses a large number of parameters.[2] - **Requires Extensive Data for Training**: To perform well, these models need large datasets and require more memory size, which may not always be available.[2] - In chatbots, they must be intelligent and able to understand long, complex, and overlapping conversations accurately and understand the meaning behind speech, and sometimes there are obstacles to this thing depending on the algorithms used, for example,

		RNN is not good in the context of long speech, unlike the power of transformers now in the long conversations and their ability to link speech together. We will talk in more detail about the problems in algorithms later.[4] - A thorough understanding of deep learning (DL), a subset of machine learning (ML) in artificial intelligence, is necessary to create successful chatbots. DL, which is often associated with deep artificial neural networks, involves sophisticated algorithms that model data through nonlinear function transformations across multiple layers. The two main categories of deep-learning chatbots are generative and retrieval-based bots. More flexible than rule-based systems, retrieval-based bots employ classification models to identify user intent and obtain pertinent responses from a database. In contrast, generative bots generate responses based on both current and previous user interactions, displaying more human-like characteristics but often struggling with grammatical accuracy and consistency.[4]
Pretrained Models (e.g., GPT-2, DIALOGPT)	- Effective for complex conversations with fine-tuning.[2]	**- High Computational and Data Requirements**: Training and fine-tuning these models require substantial computational resources and large amounts of data. **- Significant Resources for Fine-Tuning**: Adapting pre-trained models to specific applications necessitates additional computational power and expertise.[2]
Hybrid Systems	-Combines strengths of different approaches.[2]	**- Potential Decrease in Performance**: If the combined techniques are not complementary, the overall performance may suffer. **- Complexity in Management**: Integrating and managing multiple models can be complex, requiring sophisticated algorithms and additional resources to ensure effective operation.[2]

1.6 ChatGPT vs Gemini

We cannot talk about chatbots without talking about the chatbots that have created a great revolution in the world of artificial intelligence and are a source of amazement for all people, namely Gemini and ChatGPT. Artificial intelligence has advanced significantly with the outbreak of generative AI tools such as ChatGPT developed by OpenAI, and Gemini developed by Google DeepMind. These tools are categorized as generative AI (they are pre-trained using transformers), which refers to AI tools that can generate new data by identifying pertinent trends and patterns in previously gathered data. And ChatGPT, Gemini are important and popular Generative AI tools.[5]

ChatGPT is a conversational AI model designed for natural language understanding and interaction. It has gained significant attention for its ability to generate high-quality responses efficiently. Based on the GPT (Generative Pre-trained Transformers) architecture, ChatGPT uses multi-layered transformers with attention mechanisms to process input sequences of variable lengths.

Trained on a vast amount of text data from diverse sources such as books, articles, and websites, ChatGPT can identify patterns, structures, and subtleties in human speech.

It has gained popularity for generating accurate and efficient responses, particularly with its free version, ChatGPT-3.5, and the more advanced, subscription-based GPT-4.

- **Key Features and Strengths:**

Text Generation Expertise: ChatGPT excels in generating fluent and coherent text, the model's transformer structure, including encoder-decoder layers and self-attention layers, making it highly effective for various NLP tasks.

Versatility and User-Friendliness: The model is adaptable to various user needs and offers an accessible interface, making it popular among a wide range of users, from learners to industry professionals. Established with broad global use, benefiting from diverse user feedback for optimization.

Continuous Development: Regular updates ensure that ChatGPT remains relevant and capable of handling new challenges.

Global Communication Support: With support for multiple languages, ChatGPT facilitates global communication and collaboration.

Gemini is an advanced family of large language models (LLMs) with Gemini Ultra 1.5 being the most sophisticated version. Known as Bard, Gemini is integrated into various Google Workspace products like Docs, Sheets, and Gmail, assisting with tasks such as generating text and visual data. It emphasizes enterprise-grade security and privacy, ensuring user data protection.

Gemini distinguishes itself by providing precise, accurate, and up-to-date responses by utilizing Google Search for real-time information, unlike other AI tools that rely solely on pre-existing data. It prioritizes factual correctness by rating the quality of information.

- **Key Features and Strengths:**

Multimodal Versatility: Gemini excels in processing and generating both text and visual information, making it invaluable for applications such as visual search, content creation, and more. Its ability to handle diverse types of content means it can provide comprehensive answers rather than merely presenting data.

Enhanced Language Understanding: Leveraging Google's expertise in search and machine translation, Gemini delivers precise and contextually relevant responses. Its deep understanding of context and logical reasoning allows it to offer well-organized and accurate outcomes, making it suitable for users with varying technical abilities.

Energy Efficiency: The model is computationally efficient, offering potentially faster and more economical solutions.

This energy efficiency not only enhances performance but also reduces the overall computational cost, making Gemini a more sustainable choice for large-scale deployments.

Commitment to Openness: Gemini's potential for open-source

- **Challenges and Weaknesses:**

Limited Multimodal Functionality: ChatGPT primarily focuses on text and lacks integrated handling of images and other media.
Bias and Inaccuracy: The model is susceptible to biases and inaccuracies inherent in its training data, necessitating ongoing management.
Computational Costs: Running large-scale applications with ChatGPT can be computationally expensive, potentially leading to high costs.

- **Opportunities for Growth:**

Advancement of LLM Technology: Continuous improvements in ChatGPT contribute to innovation in language understanding and AI capabilities.
Wider Access to Advanced AI: By being accessible to developers and researchers, ChatGPT encourages new applications and use cases across various fields.
Automation and Task Enhancement: The model's increasing proficiency in handling tasks complements and enhances human capabilities.
Creative Innovation: ChatGPT's text generation capabilities open up new avenues for storytelling, media, and other creative applications.
Enhanced Information Retrieval: While ChatGPT lacks real-time search capabilities, advancements in AI could improve information access and retrieval in the future.

- **Potential Threats:**

Misinformation Risks: ChatGPT, like other AI models, poses the risk of generating and spreading misinformation.
Deepfake and Synthetic Media Concerns: There is growing concern about the misuse of AI for creating misleading content.
Impact on Employment: The potential for job displacement exists as AI models like ChatGPT increasingly handle content creation and knowledge work.
Perpetuation of Biases: The model risks reinforcing biases present in its training data, which could have broader societal impacts.
Security and Privacy Issues: ChatGPT's reliance on large datasets raises concerns about privacy and security.

- **Capabilities:**

Knowledge and Fact-Checking: ChatGPT relies on its internal knowledge and ongoing updates to improve accuracy in responses.
Code Generation and Understanding: The model is strong in generating code and providing explanations, making it a useful tool for developers.
Reasoning and Problem-Solving: ChatGPT is proficient in structured reasoning and logic-based tasks.
Summarization and Information Synthesis: The model excels at creating concise summaries and synthesizing information from various text sources.

Its interface is intuitive, allowing users to easily interact with the model and obtain accurate, up-to-date, and credible text-based and visual information.
Real-Time Data Retrieval: By retrieving and analyzing data from Google Search in real time, Gemini is proficient at generating realistic and reliable content, including text, images, scripts, and code. This capability makes it a valuable tool for handling complex queries and delivering precise responses.
Continuous Improvement: Through the integration of various machine learning algorithms, Gemini continuously enhances its capacity to address complex queries, ensuring that it remains a versatile and adaptable resource.

- **Challenges and Weaknesses:**

Developmental Stage: As a newer model, Gemini might lack the extensive real-world data and refinements present in more mature models.
Complexity from Multimodality: Handling various media formats can introduce complexity that may detract from performance in text-only contexts.
Risk of Misuse: Advanced capabilities increase the potential for generating harmful or misleading content.

- **Opportunities for Growth:**

Advancement of LLM Technology: Competition with models like ChatGPT drives innovation, potentially leading to significant improvements in language understanding.
Wider Access to Advanced AI: Increasing availability through open channels can foster new applications and use cases, encouraging broader adoption.
Automation and Task Enhancement: Gemini enhances its ability to handle routine tasks, complementing human work across various domains.
Creative Innovation: The model opens up new avenues in the arts and media, such as AI-driven story creation and personalized experiences.
Enhanced Search and Information Retrieval: Integration with Google's search capabilities improves query understanding and information summarization, offering more nuanced and relevant responses.

- **Potential Threats:**

Proliferation of Misinformation: Gemini's ability to generate convincing text raises the risk of spreading misinformation.
Deepfake and Synthetic Media Risks: The model's capabilities may lead to the creation of realistic but potentially harmful media.
Impact on Employment: The potential for job displacement exists as Gemini automates knowledge work.
Perpetuation of Biases: If not carefully managed, the model risks amplifying societal biases present in its training data.
Security and Privacy Concerns: Gemini's reliance on extensive data raises issues related to privacy and security.

- **Capabilities:**

Knowledge and Fact-Checking: Gemini integrates search mechanisms and tools to provide more accurate and up-to-date information.
Code Generation and Understanding: The model supports active debugging and optimization, making it a valuable tool for developers.
Reasoning and Problem-Solving: Gemini takes a methodical approach to complex problems, enhanced by the integration of external resources.
Search and Integration: The model's capabilities are bolstered by its connection to web search and tool interactions, allowing for more comprehensive responses.
Summarization and Information Synthesis: Gemini excels in synthesizing nuanced information from a variety of sources, thanks to its integration with search capabilities.

In summary, Gemini's strengths lie in its multimodal capabilities and integration with Google's search tools, while ChatGPT excels in text generation and established user experience. Each has its weaknesses and opportunities, with threats related to misinformation, bias, and privacy concerns common to both. And ChatGPT and Gemini are powerful AI models, each with unique strengths. ChatGPT excels in natural language processing, providing detailed and context-aware responses in both English and Arabic. Gemini, on the other hand, offers advanced multimodal capabilities, processing both text and visual information, and leveraging Google's expertise for precise, real-time data retrieval in multiple languages. [5][6]

Fig 2.Gemini and chatGPT chatbots

2. Arabic and English Chatbots

Despite significant advancements in chatbot technologies, these developments are primarily centered around English and other widely spoken languages, with limited focus on the Arabic language. The number of Arabic chatbots is relatively low, and there is little documentation on the methodologies used in their development. This lack of information can hinder progress for those aiming to create Arabic-language chatbots.

Most chatbots developed so far serve specific purposes, such as healthcare services, restaurant orders, airline ticketing, and education. Retrieval-based chatbots are the most common type in existing studies, utilizing natural language processing (NLP) techniques. However, existing Arabic chatbot projects often do not target the main spoken Arabic dialects, and they are generally less advanced in applying AI technologies compared to chatbots in other languages.[1]

Conversational systems have recently gained more attention due to advancements in Large Language Models (LLMs) and Language Models for Dialogue Applications (LaMDA). Nevertheless, research in conversational AI still focuses predominantly on English. Despite Arabic being one of the most popular languages on the Internet, Arabic conversational dialogue systems have only been the subject of a small number of studies to yet.[3]

Therefore, in this section, we will talk about chatbots, especially Arabic, and about the Arabic language to delve deeper into it, the evaluation methods used in it more deeply and the techniques used in Arabic chatbots and shed light on the problems in these chatbots to develop them and eliminate them.

2.1 The Arabic Language

Arabic text can be classified into three main categories:

- **Classical Arabic:** Traditionally used in religious texts, including the Quran, this form of Arabic is often referred to as Quranic Arabic.
- **Modern Standard Arabic (MSA):** The formal language used in official contexts such as academia, law, and newspapers. MSA is not spoken as a mother tongue in the Arab world.
- **Dialectical Arabic (DA):** The spoken language that varies by region and country. Although DA was traditionally not written, it has become increasingly common in written form on social media in recent years.

Among the Arabic chatbots included in the studies, Classical Arabic was the most widely used language.[1]

While developing a chatbot generally follows similar approaches regardless of the language, Arabic presents unique challenges due to its rich morphology, multiple dialects, and orthographic ambiguity. Despite these challenges, research on Arabic chatbots has increased substantially in the past years. However, resources such as available datasets, pre-trained models, and tools remain insufficient.

Most research to date has focused on MSA rather than Arabic dialects. This focus can be attributed to MSA's formal and clear structure in both written and spoken forms, which aids in analysis. However, with the rise of social media platforms, a few recent studies have started to explore chatbots in Arabic dialects, including Gulf Arabic, Saudi, Egyptian, and Jordanian dialects.[2] There is a few chatbots

have been developed using specific Arabic dialects, such as Egyptian, Saudi, and Levantine Arabic. These chatbots are tailored to communicate in the natural spoken language of particular regions, enhancing user engagement.[1]

2.2 Chatbots Application

Arabic chatbots have been developed across various domains, demonstrating versatility and effectiveness in addressing a wide range of needs, like:

1. **Healthcare Services**:
   - **General Healthcare**: Chatbots assist with healthcare-related tasks, such as providing information, booking appointments, and offering patient support. [1]
   - **Examples**:
     - **OlloBot**: A retrieval-based chatbot that helps patients manage their health and answers medical queries, designed for a closed-domain healthcare application using AIML, which focuses on specific, domain-restricted tasks in healthcare.[3]
     - **MidoBot**: An AI-based chatbot designed to assist patients with health management and medical inquiries.[3]
2. **Restaurant Orders**: These chatbots handle restaurant orders, allowing users to place orders and make reservations through conversational interfaces.[1]
3. **Airline Ticketing**: Chatbots facilitate airline ticket booking, enabling users to book flights, check flight statuses, and manage reservations.[1]
4. **Education**:
   - **General Educational Support**: Educational chatbots help with learning by providing answers to academic queries and assisting students with their studies and for university admissions, student inquiries, and academic advising.[1][4]
   - **Examples**:
     - **Homework Assistance**: Chatbots assist students with homework, provide feedback, and answer questions.[3]
     - **TOIA (Time-Offset Interaction Application)**: A bilingual (Arabic-English) interactive human avatar dialogue system that enables cross-cultural and cross-generational sharing of narratives. Inspired by the "New Dimensions in Testimony" project, it simulates face-to-face conversations.[3]
     - Jooka, Nabiha, LANA-I for university admissions, student inquiries, and academic advising.[4]
   - This chatbots can be with Generative Conversational AI generate responses to facilitate learning or Rule-Based Chatbots with predefined rules to assist in education.[3]
5. **Mental Health Support**:
   - Chatbots provide mental health support, especially in regions where discussing mental health may be culturally sensitive.[1]
6. **FAQ Chatbots**:
   - **FAQchat**: A chatbot system that converts FAQ website content into a chatbot-friendly format, answering questions using pattern-matching template rules. It serves as an interface for frequently asked Questions on websites.[3]
7. **Immersive Storytelling Chatbots**:
   - **New Dimensions in Testimony (NDT)**: A chatbot application that allows users to converse with a virtual avatar of Holocaust survivor Pinchas Gutter. It uses advanced natural language processing to create lifelike virtual avatars based on extensive video interviews, providing authentic storytelling experiences.[3]
8. **Customer Service**: Addressing questions about admissions, classes, and university events.[4]
9. **Special Needs Support**: Assisting individuals with autism spectrum disorder.[4]
   - **Examples**:
     - LANA-I.[4]
10. **Information Retrieval**: Providing information on topics like Prophet Muhammad's biography
    - **Examples**:
      - SeerahBot.[4]
11. **Tayseer Chatbot**:
    - The Tayseer architecture utilizes advanced NLP, dialogue systems, and sentiment analysis specifically tailored for Arabic. Built on the RASA framework, Tayseer manages open-ended dialogues and structured guidance through a menu option, offering instant access to college information. It addresses queries related to technical and vocational college matters, such as admissions, trainee services, academic programs, and regulations, making it an effective tool for educational and vocational support. This highlights the versatility of RASA in developing Arabic chatbots for educational contexts.[4]
12. **Notable systems and work:**

    Notable mix of chatbot and question-answering (QA) systems specifically designed for Arabic or multilingual use cases and the research projects and deep learning models that have significantly advanced Arabic natural language processing (NLP) and question-answering capabilities:

    1. **ArabChat, Mobile ArabChat, Enhanced ArabChat**: Chatbot systems that utilize pattern matching (PM) for rule-based interactions, suggesting these are chatbots with pre-defined responses based on recognized patterns in user input.[3]
       - ArabChat was the most common chatbot in the studies[1]

2. **Botta**: A retrieval-based chatbot system employing AIML (Artificial Intelligence Markup Language), a common language for designing conversational agents, indicating it relies on pre-written responses triggered by specific queries.[3]
3. **Nabiha**: A combination of AIML and PM, making it a hybrid rule-based chatbot with both retrieval and pattern-based response generation.[3]
4. **LANA**: This system uses Short Text Similarity (STS) and PM, suggesting it retrieves responses based on similarity measures in user queries, indicating a retrieval-based chatbot.[3]
5. **Rahhal**: A rule-based system aimed at helping tourists in Saudi Arabia, likely providing predefined information relevant to tourism.[3]
6. **SOQAL**: A QA system integrating TF-IDF and multilingual transformers for open-domain question answering, combining traditional text retrieval with advanced transformer models.[3]
7. **ASLSTM**: A deep learning system using LSTMs with attention mechanisms for question retrieval, enhancing the ability to match questions with relevant answers.[3]
8. **MidoBot and Deep-based DPR systems**: These systems use deep learning techniques to improve the effectiveness of Arabic question-answering systems, likely leveraging dense passage retrieval (DPR) models to find relevant information.[3]
9. **Also, there are other applications like:** Rammas, Al-HajBot, and Theyabi.[1]

Beyond these specific examples, Arabic chatbots have also been developed for a variety of other purposes across different fields, showing the adaptability of chatbot technology to diverse applications.
We found also that, there is a significant disparity between the published research on Arabic and English chatbots, particularly in the business domain. While English chatbots often focus on diverse business-related tasks, some Arabic chatbots are developed specifically for religious purposes. However, most other domains are addressed similarly in both languages.
Additionally, a relationship between the domain and the chatbot approach is observed. For education and healthcare, retrieval-based chatbots are more prevalent, as these domains typically require chatbots to access a specific knowledge base. In contrast, chatbots in business, emotions, and open domains often employ a generative approach to create more dynamic and natural conversations.[2]

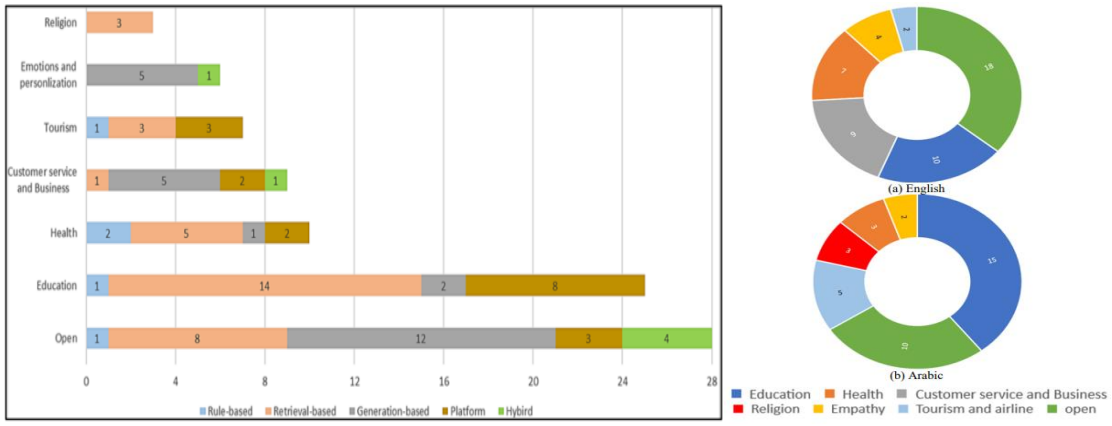

Fig 3. Finding in Domains with Approaches Perspective and with with Languages Perspective.[2]

2.3 Evaluations techniques in Arabic chatbots

We need to evaluate our model's behaviors with the new inputs and data from users, so we need evaluation metrics to quantify the performance of machine learning models on test data. These equations and metrics are applied to evaluate the quality, relevance, and effectiveness of chatbot responses in both Arabic and English, depending on the specific characteristics and needs of each language. So the way of evaluation can differ depending on the used language. The evaluation of chatbots in both Arabic and English involves a mix of traditional NLP metrics, ranking-based approaches, and specialized metrics tailored to the linguistic characteristics of each language. While automated metrics offer efficiency, human-based evaluations, such as user satisfaction and empathy, remain crucial for accurately assessing chatbot performance. Arabic chatbots often rely more on human evaluations due to challenges like the lack of available data resources, while English chatbots commonly use metrics like BLEU and ROUGE.[2]

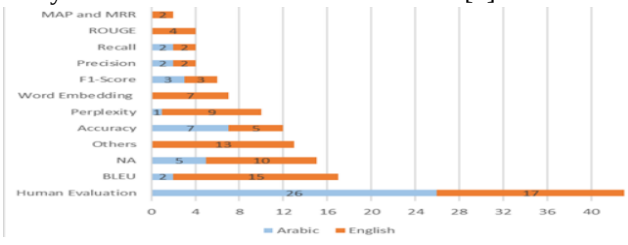

Fig 4.Finding in Metrics Perspective [2]

So there are important differences in metrics between Arabic and English chatbots, and from those Metrics:

Human-Based Metrics:

Human evaluation involves having a group of people communicate with the chatbot and evaluating various aspects using evaluation frameworks or questionnaires.[2]

1. **User Satisfaction:**
   - **Arabic and English Chatbots:** User satisfaction is a common metric for evaluating chatbots in both languages. It typically involves gathering feedback from users to assess how well the chatbot meets their needs and expectations. This metric is qualitative and depends on user responses.
   - **Examples:** A feedback questionnaire or survey could ask users to rate their satisfaction on a scale from 1 to 5, and sentiment analysis can be applied to analyze the sentiment of user comments.[2]

Automatic-Based Metrics:

1. **Traditional Evaluation Metrics:**
   - **Precision, Recall, and F1-Score:**
     - **Arabic and English Chatbots:** These metrics are universally applied in NLP tasks to measure the accuracy, relevance, and quality of the chatbot's responses.[2][3][4]
     - $$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$
     - $$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$
     - $$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$
   - **Accuracy:**
     - **Arabic and English Chatbots:** This metric is used to evaluate the correctness of the chatbot's responses against a gold standard.[2][3][4]
     - $$\text{Accuracy} = \frac{\text{True Positives (TP)} + \text{True Negatives (TN)}}{\text{Total Number of Predictions}}$$

     Where: TP = True Positives, TN = True Negatives, FP = False Positives, FN = False Negatives
   - **Exact Match (EM):**
     - **English Chatbots:** Commonly used in English QA systems to measure how many predicted answers match the reference answers exactly.[3]
     - $$\text{EM} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$
2. **Ranking-Based Metrics:**
   - **C@1 (Correct at 1):**
     - **Arabic and English Chatbots:** This metric evaluates whether the chatbot provides a correct answer or opts not to answer when unsure. It is useful for assessing the system's ability to handle uncertain situations.[3]
     - $$C@1 = \frac{1}{\text{TQ}} \sum_{i=1}^{\text{TQ}} \frac{1}{\text{Rank}(A_i)}$$

       Where TQ is the total number of questions, and Rank(Ai) is the rank of the first correct answer.
   - **Mean Average Precision (MAP):**
     - **Arabic and English Chatbots:** Used in ranking tasks to evaluate how well the chatbot ranks relevant responses higher than irrelevant ones.[2][3]
     - $$\text{MAP} = \frac{1}{Q} \sum_{i=1}^{Q} \frac{\sum_{k=1}^{n} P(k) \times \text{rel}(k)}{\text{Number of Relevant Documents}}$$

       Where Q is the number of queries, P(k) is the precision at position k, and rel(k) is a binary indicator of whether the document at position k is relevant.
   - **Mean Reciprocal Rank (MRR):**
     - **Arabic and English Chatbots:** Assesses the quality of the first correct answer's rank, measuring how quickly the correct answer appears in the list of responses.[2][3]
     - $$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}$$

       Where rank is the rank of the first relevant document for the i query.
3. **Textual and Embedding-Based Metrics:**

- **ROUGE:**
  - **Arabic and English Chatbots:** Used in text summarization tasks to measure the overlap between the generated text and reference text. Common variants include ROUGE-1 (unigrams) and ROUGE-2 (bigrams).[2,3]

$$\text{ROUGE-N} = \frac{\sum_{\text{match} \in \text{n-gram}} \text{Count}_{\text{match}}(n\text{-gram})}{\sum_{\text{n-gram} \in \text{reference}} \text{Count}_{\text{reference}}(n\text{-gram})}$$

- **BLEU:**
  - **English Chatbots:** Widely used for machine translation and generative chatbot responses, measuring the n-gram overlap between the chatbot's response and reference text.[2][3]

$$\text{BLEU} = \text{BP} \times \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

Where BP is the brevity penalty, Wn is the weight of each n-gram, and Pn is the precision of n-grams.

- **BERTScore and MoverScore:**
  - **Arabic and English Chatbots:** Embedding-based metrics used to evaluate the semantic similarity between the chatbot's output and reference answers.[3]
- **Automated Error Analysis Tools (e.g., AMEANA):**
  - **Arabic Chatbots:** Tools like AMEANA are developed specifically for Arabic to identify and analyze morphological, syntactic, and semantic errors in chatbot responses.[3]

Metrics Unique to Arabic Chatbots:

1. **AL-BLEU:**
   - **Arabic Chatbots:** A metric tailored specifically for Arabic, considering the linguistic features unique to the language.[3]
2. **Morphologically-Enriched Embedding Metrics:**
   - **Arabic Chatbots:** Designed to handle the morphological richness of Arabic, these metrics are important for accurately evaluating Arabic chatbots.[3]

Evaluation Methodologies and Tools:

1. **User Feedback:**
   - **Arabic and English Chatbots:** Collecting user feedback through questionnaires and sentiment analysis is common in both languages. This feedback helps refine and improve chatbot functionality.[4]
2. **Error Handling and Fallback Mechanisms:**
   - **Arabic and English Chatbots:** Evaluating how well the chatbot handles invalid or non-existent inputs is crucial for both languages, ensuring the chatbot can guide users back to valid options.[4]

There are other many ways for evaluation used in both Arabic and English chatbots, like Confusion Matrix or calculating the Response Time on the user inputs, and bot responses are logged to ensure promptness or assess the scalability of the response and more and all of them help us to know how much the response fit the user needs and serve him to improve the user experience and the effectiveness of chatbots in both Arabic and English contexts.[4]

2.4 Used Tools

There are a lot of tools that help in building chatbots like platforms, programming languages, or NLP tools. Modern platforms or programming languages are used to create a variety of chatbots for various uses, such as C# in Verbot 5.0, Snatchbot, Microsoft Visual Studio connected to the Google Translate API, and the Twilio platform, and a few Python libraries are used, including chatterbot and chatterbot_corpus. Used to build Arabic or English chatbots or chatbots also by other languages. [2]

- **Platforms and Frameworks:**

These platforms remove the need for coding expertise to build chatbots from the ground up. They streamline the development process and standardize certain implementation steps. While some platforms, like Pandora, offer a clear approach using a retrieval-based method, many others obscure the inner workings of their systems. Google DialogFlow, Facebook Wit.ai, IBM Watson Conversation, and Microsoft Azure are cloud-based platforms supporting various programming languages in addition to natural languages. However, they differ considerably in other respects. From platforms and frameworks that are used in different languages and their problems: [2]

1. **Google's DialogFlow**: Supports over 45 languages except Arabic and has limitations in handling synonyms and documentation quality.[2]
2. **Pandorabots**: Supports Arabic but struggles with Arabic spelling errors.
3. **IBM Watson Conversation**: Supports 13 languages, including Arabic, but lacks enhanced intent detection and misspelling autocorrection.[2]
4. **Microsoft Azure**: Supports over 100 languages but has issues with error rates in translation-based replies.[2]

5. **RASA Framework**: Open-source, supports any language, with a focus on customization for intent recognition and dialogue management. Not support Arabic predefined trained entities. Including RASA NLU for language understanding and RASA Core for dialogue management.[2][3][4]

RASA Framework	RASA NLU Pipeline Components
○ **RASA NLU**: For natural language understanding, including intent and entity recognition. ○ **RASA Core**: Manages conversation flow using machine learning. ○ **Domain Definition**: Includes intents, entities, and actions tailored to specific queries. ○ **Rules and Stories**: Establish conversational guidelines and example paths. ○ **Actions**: Custom and predefined responses, API calls, or database queries.	○ **WhitespaceTokenizer**: For simple text segmentation. ○ **RegexFeaturizer**: Identifies patterns in text. ○ **LexicalSyntacticFeaturizer**: Captures lexical and syntactic features. ○ **CountVectorsFeaturizer**: Converts text to numerical data using bag-of-words and character models. ○ **DIETClassifier**: For intent and entity recognition using transformer technology. ○ **EntitySynonymMapper**: Aligns terms for consistency. ○ **ResponseSelector**: Selects appropriate responses from predefined options.

6. **Facebook Bot Engine (Wit.ai)**: It supports over 100 languages, but training for Arabic is time-consuming to understand all forms of Arabic text.[2]
7. **Chatfuel**: Supports multiple languages, including Arabic, but is inflexible in conversation flows.[2]
8. **Recast.AI**: Supports over 15 languages, including Arabic. It also has the documentation is poor.[2]

- **Arabic NLP Tools:**

Also, there are a lot of tools in NLP that are designed to handle the unique challenges posed by the Arabic language and for other languages too. The Arabic NLP tools and resources are invaluable for researchers and developers working in Arabic NLP to enhance language processing, offering specialized functionalities to tackle the complexities of Arabic language processing. Those NLP tools are used to make machines understand the user text. They are used in many applications that require text understanding like intent recognition, and topic modeling, and also help us build better chatbots. They can help us in different ways like helping us to normalize, cleaning the text, classifying it also, and in many different other ways. From that NLP Arabic tools and their usages:[3]

1. **MADAMIRA**

   Description: MADAMIRA is an Arabic morphological text-analysis tool that combines the functionalities of two widely-used systems, MADA and AMIRA. It offers comprehensive tools for pre-processing Arabic text.

   Key Features:
   - Tokenization: Segmenting input text into individual units.
   - Lemmatization: Reducing words to their base or root form.
   - Part-of-Speech (POS) Tagging: Assigning grammatical categories to tokens.
   - Base Phrase Chunking (BPC): Identifying phrases and their syntactic roles.
   - Named Entity Recognition (NER): Identifying and extracting named entities like people, locations, and organizations.[3]

2. **Farasa**

   Description: Farasa is an Arabic segmenter that uses SVM-rank to learn feature vectors for word segmentation. It includes two variants: FarasaBase and FarasaLookup.

   Key Features:
   - Dialectal Analysis: Handling variations in Arabic dialects.
   - Diacritization: Adding diacritical marks to text.
   - Text Normalization: Standardizing text format.
   - POS Tagging: Assigning grammatical categories.
   - NER: Identifying and extracting named entities.
   - Machine Translation (MT): Demonstrates high accuracy in translating text.[3]

3. **CAMeLTools**

   Description: CAMeLTools is a comprehensive toolkit for Arabic natural language processing tasks, developed to address limitations in previous rule-based systems.

   Key Features:
   - Transliteration: Converting text from one script to another.
   - Orthographic Normalization: Standardizing the writing of words.
   - Discretization: Simplifying complex linguistic data.

- Dialect Identification: Detecting different Arabic dialects.
- Morphological Modeling: Analyzing word forms and structures.
- Sentiment Analysis: Detecting emotions or opinions in text.
- NER: Identifying and extracting named entities.
- Text Classification and Clustering: Grouping and categorizing text data.[3]

4. **Masader+**

Description: Masader+ is an online catalog of Arabic NLP data, providing a comprehensive repository of linguistic resources and datasets.

Key Features:

- Offers datasets for tasks like NER, question-answering, and sentiment analysis.
- Provides detailed information about each dataset, including size, format, and suitability for specific tasks.
- Includes instructions for data downloading and usage, along with links to related resources and publications.
- **Special Interest Group on Arabic NLP (SIGARAB)**

Description: SIGARAB is a professional organization under the Association for Computational Linguistics (ACL) that promotes the growth of Arabic NLP technologies.

Key Features:

- Fosters knowledge exchange among researchers and practitioners.
- Organizes regular meetings, conferences, newsletters, and proceedings.
- Provides diverse resources such as datasets, software, and documentation for Arabic NLP research and development.[3]

5. **MADA Tool**:
   - Provides morphological analysis and disambiguation for Arabic text.[4]
6. **Pre-trained Models**:
   - Embedding models like **BERT**, **GloVe**, **AraBERT**, and **AraGPT2** are used for enhanced NLP capabilities in Arabic and other languages.[4]

There are a lot of tools that help us, the idea of naming it by tools that they all ways and kits for trying to build better chatbots in understanding the user and generate relevant answers. Also, we can Standardize text format, including lowercasing and removing irrelevant characters, and tokenizing the text by NLP or manually or by pattern matching and regular expressions all different ways and tools can help in achieving the same purpose here which is data preprocessing.

From that, we can say that the tools are the kits that help us build chatbots from scratch to the end including the data collection that can happen in different ways like Existing documentation, FAQs, expert knowledge, web scraping, surveys, or from websites that have different organized formatted data as Kaggle or Github.

There are tools like we saw from language, data processing, and cleaning. Also, there are tools for integration and deployment to make our chatbot work on websites or phones, and desktop apps, like Using Next.js for user interface in Front-End Development, Nest.js for managing functions like user authentication in Back-End and Microsoft Azure for deployment, and other also for hosting.[4]

As we saw also there are the programming languages and frameworks that help us in building from the first step to the end and the platforms that help the business people without involve with the complexity of programming.

2.5 More used techniques

These techniques as we said before enable chatbots to understand user inputs but here we will highlight the diverse potential of Arabic chatbots while also indicating areas where further development and innovation are needed to improve their effectiveness and usability.[1]

The integration of various techniques and approaches in building Arabic chatbots highlights the progress made in enhancing their effectiveness, while also showing areas that need further innovation. Here's an organized breakdown of the techniques and tools used in Arabic chatbot development, merging the provided concepts:

Traditional Techniques

1. **Pattern Matching (PM)**
   - **Overview**: The majority of Arabic chatbots still rely on pattern-matching techniques, where predefined patterns are matched with user inputs to generate responses.
   - **Advantages**: Computationally inexpensive and easy to implement.
   - **Limitations**: Lacks flexibility and sophistication, struggles with complex queries, and cannot adapt to unseen inputs.[1]
2. **Rule-based Systems**

- **Examples**: Systems like **Rahhal** or **Nabiha** use rule-based interactions to deliver responses, often for specific, structured domains like tourism or healthcare.
- **Limitations**: Rule-based systems require manually crafted rules, making them inflexible and prone to error in diverse language contexts such as Arabic dialects.[3]

Advanced Natural Language Processing (NLP)

1. **NLP Techniques**

   A limited number of Arabic chatbots use NLP techniques, which involve more complex processing of natural language inputs to enhance understanding and response generation, enabling more human-like interactions and handling a broader range of user queries. It can help in a lot of ways, used for tasks such as named entity recognition, intent classification, and entity extraction, enhancing a chatbot's ability to understand a broader range of queries. Like also do Morphological and Syntactic Disambiguation, and it's like part-of-speech tagging, stemming, and lemmatization are used to handle the complexities of Arabic.[1][4]
   - **Morphological and Syntactic Disambiguation**: NLP techniques, such as part-of-speech tagging, stemming, and lemmatization, are essential for handling Arabic's rich morphology and complex syntax.
   - **Tools**:
     - **Pre-trained Models**: AraBERT, BERT, and AraELECTRA. And **AraBERT** and **AraELECTRA**: are pre-trained transformers models are specifically designed for Arabic text and excel in various NLP tasks, including question-answering and intent recognition. Their ability to capture contextual meaning makes them highly effective in Arabic chatbot development.[3]
     - **Embedding Models**: Word2Vec, GloVe, and FastText (static embeddings) that provide foundational word representations but lack context sensitivity. And ELMo, BERT (contextualized embeddings) capture more nuanced semantic meanings by considering the context of words within sentences.[3]
   - **Applications**: Used for tasks such as named entity recognition, intent classification, and entity extraction, enhancing a chatbot's ability to understand a broader range of queries.
2. **Dual Intent and Entity Transformer (DIET) Architecture**
   - **Overview**: A transformer-based model used for intent classification and entity extraction, is crucial for processing complex user inputs.
   - **Integration**: DIET can be combined with pre-trained embeddings like BERT, GloVe, and ConveRT to improve intent recognition in Arabic chatbots like the "Tayseer" chatbot. And DIET is compatible with models like **AraBERT** and **BERT**, allowing the chatbot to better handle Arabic text by capturing semantic meaning from user inputs.
   - **Focus on Attention Mechanisms**: Helps the model focus on relevant parts of the input, especially useful for languages with complex syntax, like Arabic.[4]

Deep Learning-Based Approaches

1. **Generative-Based Models**
   - **Overview**: Though less common in Arabic chatbots, generative models like sequence-to-sequence (Seq2Seq), Long Short-Term Memory (LSTM), and Generative Adversarial Networks (GANs) create responses dynamically, without predefined patterns. These models may incorporate techniques such as supervised learning, unsupervised learning, reinforcement learning, and adversarial learning.[1]
   - **Models**: BERT, GPT, LSTMs, and their variants such as **AraBERT**, **AraELECTRA**, and **ASLSTM**.
   - **Challenges**: Developing comprehensive datasets for Arabic, including dialect variations, remains a challenge.[4]
2. **Transformers and Large Language Models (LLMs)**
   - **Overview**: Transformers like BERT, GPT, and newer LLMs (e.g., GPT-3, LaMDA) utilize self-attention mechanisms to handle long-range dependencies, making them highly effective for complex conversational AI tasks and contextual understanding.[3]
   - Deep learning-based approaches, such as sequence-to-sequence models and transformer-based models, are used for generating responses and understanding Arabic text.[4]
   - **Applications**: These models are used in question-answering systems like **SOQAL**, and pre-trained models like **AraBERT** and **AraELECTRA** show significant improvements in Arabic NLP tasks.

Hybrid Approaches and Custom Innovations

1. **Hybrid Models**
   - **Overview**: Combining retrieval-based methods with generative models to create more flexible and accurate responses.
   - **Example**: Chatbots like **Nabiha** use both AIML and pattern matching to improve response accuracy in mixed retrieval and rule-based systems.

- **Customized Knowledge Representation**: In chatbots like **Tayseer**, custom knowledge representations are tailored for Arabic syntax and morphology, enhancing the understanding and processing of Arabic language inputs.[4]

2. **Feedback Loops and Sentiment Analysis**
   - **Overview**: Some systems incorporate sentiment analysis to improve user interactions over time, such as adjusting responses based on user satisfaction and emotional tone. This helps fine-tune the chatbot's responses in a continuous feedback loop.[4]
3. **Integration with Existing Technologies**
   - **Overview**: Some Arabic chatbots have been adapted from existing conversational agents like ALICE or use services like IBM Watson Conversation to leverage advanced AI technologies. However, the integration of such technologies in Arabic chatbots remains limited.[1]

So in the end the transition from rule-based and retrieval-based systems to deep learning-based approaches has brought significant advancements in Arabic natural language processing (NLP), particularly in chatbot development. Deep learning models offer enhanced accuracy and a deeper understanding of context, overcoming many limitations of traditional methods. However, challenges persist, especially in constructing comprehensive and representative datasets for Arabic, as well as managing the complexities of various Arabic dialects and linguistic variations.[3]

Within frameworks like RASA, deep learning models have been integrated with custom Arabic NLP pipelines for tasks such as intent classification, entity extraction, and response retrieval. This contributes to creating robust, effective, and user-friendly Arabic chatbots, especially for domains like higher education.[4]

Despite the progress made with deep learning models like LSTMs and transformers, generative models remain underutilized in Arabic chatbot development. While systems like ASLSTM and MidoBot show promise, the adoption of generative approaches in Arabic NLP is still limited. One of the biggest obstacles in building effective Arabic chatbots is the lack of large, high-quality datasets that represent the diversity of the Arabic language, including its dialects. This dataset scarcity affects the performance of both retrieval-based and generative models.

Historically, Arabic chatbot research has been dominated by retrieval-based approaches, with techniques like pattern matching and AIML being the most common. However, since 2018, there has been a growing interest in generation-based methods, with LSTM, seq2seq, and GRU emerging as popular techniques. Still, compared to English-language chatbot research, Arabic chatbot development lags, particularly in the use of generative models. Therefore, more focus on enhancing Arabic datasets and exploring generation-based models could unlock further potential in Arabic chatbot development.[2]

2.6 Limitations in Arabic Chatbots

There are unique challenges faced in developing effective and culturally relevant chatbots for the Arabic-speaking world. It is also associated with developing Arabic chatbots and leveraging deep learning algorithms in this context. These limitations highlight the need for more focused research, better resources, and more innovative approaches to advance the development of Arabic chatbots to overcome these challenges and also highlight areas where improvements can be made to enhance the effectiveness and accuracy of Arabic chatbots. Addressing these issues involves refining intent classification, optimizing response times, expanding FAQ coverage, and improving sentiment analysis methods tailored to the Arabic language.[1][2][4][26] Here are some general limitations on Arabic chatbots discussed in the papers:

Limitations in Developing Arabic Chatbots and Addressing the Arabic Language:

1. **Limited Research and Development:**
   - **Scarcity of Arabic Chatbots**: There is a significant gap in the availability of Arabic-language chatbots compared to English-language ones. The limited research and resources hinder the development of sophisticated Arabic chatbots. This gap is particularly evident in the development and adoption of generation-based approaches, which only started gaining attention in Arabic chatbot research around 2018. [1][3]
   - **Predominance of Specific Techniques and Technology Limitations**: Many Arabic chatbots rely on rule-based or retrieval-based methods, traditional rule-based systems often rely on pattern matching (PM) and AIML (Artificial Intelligence Markup Language), which can be limited in handling the complexity of Arabic, restrict their ability to generate diverse and contextually appropriate responses and limiting their flexibility and adaptability compared to more advanced AI systems. More advanced techniques such as Natural Language Processing (NLP) and generative-based models. LSTM is the most employed technique, followed by seq2seq and GRU. These approaches may limit the diversity and innovation in developing Arabic chatbots. Deep-learning techniques extensively explored in English dialogue systems require further improvement for Arabic [1][3][4][26]
   - **Insufficient Research on Deep Learning**: The application of deep learning algorithms to Arabic chatbots is underexplored, which hampers the advancement of Arabic NLP technologies, while some works have demonstrated the potential of neural models in Arabic language understanding, such as machine transliteration and stemming, there is still limited research on

applying deep learning algorithms to Arabic chatbots. This lack of research hinders the advancement of Arabic NLP technologies.[1]

- **Limited Coverage of the Range of FAQ Topics:** The chatbot's ability to address a wide range of FAQ topics is limited. Some users found the coverage insufficient, highlighting the need for a broader range of topics.[4]

2. **Complexity of the Arabic Language:**
   - **Linguistic Complexity**: Arabic's rich morphology, orthographic variations, multiple dialects, syntactic variability that can complicate sentence parsing and understanding, syntax and orthographic too ambiguity pose significant challenges in natural language processing (NLP) and limited resources for training and evaluation. The language's complexity complicates the development of accurate and robust chatbots in accurately interpreting and generating responses. The language's complexity adds difficulties in processing and understanding natural language inputs.[1,4,26]
   - **Dialectal Variations**: Arabic is spoken in numerous dialects, each with its unique characteristics. Most chatbots focus on Modern Standard Arabic (MSA), which may not fully represent the spoken language in everyday conversations. This variation complicates understanding and generating accurate responses. There is a lack of focus on dialectal Arabic (DA), which is the natural spoken language in everyday conversations that is increasingly relevant due to the rise of social media platforms. And also regional dialects and variations can affect the accuracy of language processing and response generation.[1][3][4]
   - **Script Nuances and Morphological Variations**: Variations in script and morphological complexity, including prefixes, suffixes, and root patterns, add difficulty in processing and interpreting Arabic text. Also, Arabic words often have multiple forms and meanings, making it difficult for chatbots to disambiguate correctly. Ensuring responses are grammatically correct and idiomatic in Arabic requires careful attention to language nuances.[4]
3. **Data and Resource Limitations:**
   - **Lack of Accessible Datasets**: The limited availability of high-quality, annotated datasets in Arabic makes it challenging to train and optimize AI algorithms for the language. This scarcity impacts the performance and generalization ability of chatbot models.[1][3][4][26]
   - **Training Data**: Building a comprehensive dataset for training may be more challenging due to the need for diverse and representative examples in Arabic.[4]
   - **Standardization and Tokenization Challenges**: Effective text normalization and tokenization are complex due to Arabic's diverse linguistic features and unique word formation, which affects the accuracy of text processing.[4]
   - **Resource Limitations**: The development of Arabic chatbots is constrained by limited resources, such as data and linguistic tools, which are necessary for training and improving these systems. This scarcity of resources limits the ability to create sophisticated and effective chatbots for Arabic-speaking users.[3]
   - **Library and Tool Support**: There is a shortage of specialized libraries and tools tailored to Arabic, which restricts the development of effective chatbots.[4]
4. **Technological and Performance Challenges:**
   - **Performance Issues**: Arabic chatbots, particularly those using simple models, may struggle to meet the sophisticated needs of users. Issues like longer response times for specific intents and performance degradation under high volume highlight areas where optimization is needed.[4]
   - **Reliability and Accuracy**: There are concerns about the reliability and accuracy of chatbots when integrated on a wide scale in educational settings. This challenge is particularly significant during the implementation of chatbots in learning processes.[3]
   - **Sentiment Analysis Limitations**: The sentiment analysis models, often lexicon-based, may not accurately interpret feedback due to the nuanced nature of the Arabic language, especially with the lexicon-based approach which relies on predefined sentiment scores and might not cover all linguistic nuances and variations in Arabic. [4]
   - **Scalability and Response Time**: The scalability of Arabic chatbots is limited, as performance tends to degrade with an increasing number of concurrent users, and handling larger volumes may still impact performance and response times. Some intents, such have longer response times, indicate areas where performance could be optimized.[3][4]
5. **Cultural Sensitivity and Contextual Understanding:**
   - **Cultural Sensitivity**: Arabic chatbots need to be culturally sensitive, avoiding offensive or inappropriate responses, especially when addressing topics like mental health, which may be considered taboo in certain regions. The lack of multicultural representation in AI development can result in biases.[1][3]
   - **Context Sensitivity**: Recognizing intents and entities accurately in Arabic requires context-aware processing, which is challenging given the language's morphological richness. And the chatbot occasionally misclassifies user inputs due to similar intents, as shown by errors in predicting intents like “user_ask_major_change” versus “user_ask_about_conditions_changing_specialization”. This results in responses that do not align with the user's intended query. [3][4]

- **Limited Support for Emotion and Context**: Arabic chatbots often lack emotional intelligence and the ability to engage in human-like conversation. The research on incorporating emotions, contextual cues, and natural language understanding is still underdeveloped in the context of Arabic chatbots.[3]

6. **Evaluation and Research Gaps:**
   - **Insufficient Research**: There is a lack of focused research on developing advanced, deep learning-based conversational models for Arabic. This gap slows down the progress in developing more sophisticated chatbots that leverage machine learning and word embedding. Arabic chatbots are relatively underrepresented in the field of natural language processing (NLP) compared to English chatbots. This lack of research and resources poses a challenge for developing effective Arabic chatbots. The literature on Arabic chatbots is sparse, with few studies available that explore methodologies, technologies, and applications specific to the Arabic language. This lack of research can slow down progress in the field.[1][3][26]
   - **Limited Evaluation Metrics**: The evaluation of Arabic chatbots often relies on human judgment, as standardized metrics like BLEU, commonly used for English chatbots, are not well established for Arabic. This reliance on human evaluation can introduce subjectivity and hinder the development of automated evaluation techniques.[3]
7. **NLP Limitations**:
   - **Limited NLP Tools**: Although tools like Farasa and CAMeL exist to handle Arabic's linguistic features, the overall development of Arabic conversational AI is hindered by the lack of advanced machine-learning algorithms and natural-language understanding models tailored to Arabic.[3]
8. **Future Directions for Improvement:**
   - **Focus on Dialectal Task-Oriented Dialogue**: More research and development should focus on creating datasets and conversational systems that cater to specific Arabic dialects.[3]
   - **Advanced Word Embeddings and Machine Learning**: Emphasizing the development of efficient Arabic conversational systems using advanced word embeddings and machine-learning methods can address challenges like dialectal diversity and resource scarcity.[3]
9. **Arabic-Specific Challenges**
   - **Arabic Varieties**: Arabic has three main varieties—Classical Arabic, Modern Standard Arabic (MSA), and colloquial dialects. Each variety has its unique characteristics, and the diversity of dialects across regions makes it difficult to build a chatbot that can communicate effectively with all Arabic speakers.[3]
   - **Ambiguity and Orthography**: The omission of diacritical marks, common in MSA, leads to lexical ambiguity, complicating the interpretation and processing of text. Arabic orthography allows for optional diacritical marks, leading to ambiguity. The lack of consistent use of these marks adds complexity to natural-language processing.[3]
   - **Morphological Complexity and Tokenization**: The intricate rich morphology of Arabic, with its numerous inflectional forms, poses challenges in developing robust NLP systems. Arabic language morphology and tokenization are complex, which can impact the accuracy of text processing and sentiment analysis. Arabic words can have multiple forms, making it challenging to tokenize and analyze accurately.[3][4]
   - **The Arabizi Phenomenon**: Arabizi, a mix of Arabic and Latin characters, introduces additional complexity in processing text, as it lacks standardization and varies significantly in how it is written.[3]
   - **No Capital Letters and Code Switching**: The absence of capital letters in Arabic complicates named entity recognition. Additionally, Arabic speakers often code-switch between languages, making it challenging for chatbots to accurately process and respond to multilingual queries.[3]
10. **Limitations in Voiced Chatbots:**
    - **Speech Recognition and Synthesis**: Compared to English, Arabic lacks advanced Text-to-Speech (TTS) and Automatic Speech Recognition (ASR) systems. This is due to the complex phonetic system, making accurate speech recognition difficult, especially with various dialects and accents.[3]

In summary, developing Arabic chatbots faces challenges due to the language's complexity, limited resources, and underexplored deep learning applications. The rich morphology, dialectal variations, and orthographic diversity of Arabic make natural language processing (NLP) tasks difficult, leading to limitations in intent recognition, sentiment analysis, and response generation.

Most Arabic chatbots rely on rule-based methods, lacking the flexibility of more advanced AI systems. Performance issues, slower response times, and limited support for dialects further hinder development. Additionally, the scarcity of high-quality datasets and specialized tools restricts progress.

To overcome these challenges, more research and resources are needed to improve deep learning applications, create better tools, and develop richer datasets tailored to the complexities of the Arabic language. This will enhance the accuracy, scalability, and effectiveness of Arabic chatbots.

3. Intelligent Chatbots

As is well known, artificial intelligence (AI) is the process by which machines are made to simulate human intelligence to carry out tasks that normally call for human intelligence. This encompasses various processes such as learning, reasoning, problem-solving, perception, understanding natural language, and decision-making.[31]
The chatbots are the same which try to act like humans, understand human language, and give responses like humans, Meaningful and relevant. We use AI to try to make the machines understand the user's text before answering for better responses than any unintelligent chatbots.
The rapid technological advancements of AI aid in achieving this goal and help in the development of more flexible chatbots that can produce human-like conversation.[2]
In this section, we will talk about different Algorithms in machine and deep learning that were used in different papers to build intelligent Chatbots, limitations in algorithms, chatbots that affected results, and different implementations in building better accuracy and more smart chatbots. In the previous sections, we talked about the different techniques for building chatbots and Arabic and English chatbots. In this section, we will focus on smart chatbots built using artificial intelligence techniques. We will review some of the most important algorithms used to build them through different papers, as well as the advantages and disadvantages of each algorithm.

3.1 Implementation way for Chatbots

Implementing a chatbot varies from one chat to another depending on the chatbot, what it will be used for, and the language used in it. Implementing a chatbot is not an easy thing. Making it understand human language and answering the user with an appropriate answer is a big challenge. Therefore, before implementing some chatbots, some people try to understand the purpose for which the chatbot is used clearly and use the appropriate data to build it and try to understand the data, prepare it, make Preprocessing like by cleaning it removing punctuation, extra symbols, and other unnecessary elements from the dataset to ensure it is suitable for training the chatbot and make Feature Extraction by different techniques to help in preparing the data for training the chatbot before choosing the algorithm and start training our Chatbot by it.
Also, some use different additional techniques to try to enhance the quality and diversity of generated responses like Beam Search Decoding and others. Beam search decoding is employed during the inference phase. Beam search is a heuristic search algorithm used for generating sequences of words in chatbot responses. It explores multiple potential response sequences and selects the most likely ones based on predefined parameters.
In addition, hyperparameter tuning can affect our model accuracy, and choosing the appropriate method to test and evaluate our model by different metrics is also important in evaluating our Chatbot strength. We will talk about all these methods that communicating together to try to enhance our chatbots that can be used in our implementation.
The implementation methodology described in the research involves a combination of preprocessing techniques, deep learning models, and heuristic search algorithms to develop a functional chatbot capable of interacting with users using natural language.[16]

3.1.1 Choose and collect data:

Choosing the right data is the most important step. Choosing the data that our chatbot will learn from and respond based on is one of the most important steps. We must accurately define the topic of the chatbot and build its data accurately. We can search for this data or build it through surveys or request it from some organizations if it exists but is closed within the organizations or web scrap as well for websites if it's possible. From sources for data Corpus datasets, Kaggle, Github, and different websites and platforms like Twitter platform or Universities and institutes website.
Data must also be organized in an appropriate format can Machine or deep learning models deal with and label it. The various types of data, including:

- **Structured Data**: Data organized into tables or databases with a fixed schema, such as numerical, categorical, or ordinal data.
- **Unstructured Data**: Data that lacks a predefined data model or structure, such as text, images, audio, and video.
- **Semi-Structured Data**: Data that has some structure but does not conform to the strict structure of relational databases, such as JSON, XML, and log files.

The size of the data is also important especially if the training with Deep learning algorithms needs a big size of data for good learning unlike machine learning, sometimes increasing the data beyond a certain limit does not help improve accuracy, and accuracy reaches a certain limit and stops growing and stabilizes.
Gather relevant data from various sources, ensuring it is labeled and representative of the problem domain to facilitate effective training..

3.1.2 Prepare and preprocess Dataset

The collected data undergoes preprocessing steps to clean and prepare. Data must be prepared and cleaned before training and data preprocessing techniques applied to this, to ensure that the data is well-prepared for model training and to simplify the text. Excess data and unwanted information in models as in chatbots can give wrong and unwanted information. Bad data makes a bad model, so this is the most important step of all the steps. The chatbot utilizes NLP using Natural Language Understanding (NLU) techniques such as tokenization, part-of-speech tagging, stop-word removal, stemming, and entity extraction are applied to preprocess user queries and to analyze and categorize input sentences.

So first the data must be built clean, good-preprocessed and organized in a way that the model can learn from, and from the methods and steps to do this are: [19][23][26][27][30]

Cleaning & Normalization:

- Remove noise, handle missing values, punctuation, special symbols, emails, HTML tags, accented characters, non-alphanumeric content, extra whitespace, and other unnecessary elements to ensure data uniformity, consistency, and relevance and are suitable for training the chatbot.[16][18][21][22]

Stopwords Removal:

- Eliminate commonly used words that don't contribute meaningfully to the analysis, such as prepositions and conjunctions.[29]
- Suffer from potential loss of context or meaning, especially in languages with rich morphology or complex syntax.[29]

Tokenization:

- Split text into individual tokens (words or subwords) to simplify tasks like parsing and text analysis by converting text into smaller units for analysis and further processing using tools like Keras or Taqseem for Arabic language tokenization.
    - **Taqseem**: A tokenizer specific to the Arabic language. Its limitations in handling specific linguistic nuances or variations within the Arabic language are not discussed.
    - **Tf-idf**: Term Frequency-Inverse Document Frequency, a common technique used for tokenization and feature extraction in natural language processing. It's not capture semantic meaning or context effectively.
    - **WhiteSpace Tokenizer**: Tokenization based on white spaces, which may not handle languages with complex word formations or agglutinative languages effectively.
    - **Arcab**: A data-driven, unsupervised method for tokenizing subwords in Arabic phrases, and potential limitations related to its performance on different types of text or languages are not discussed.
    - **ConveRT**: A compact model of neural response selection for dialogue using transformer-backed dual-encoder networks, also any potential limitations related to its scalability or performance on Arabic text are not mentioned.[18][25][29]

Stemming & Lemmatization:

- Apply techniques like stemming and lemmatization to reduce words to their base forms, simplifying the data.

Padding and Encoding:

- Convert tokenized text into numeric sequences and apply post-padding to ensure uniform input lengths across the dataset.[18]

Keyword Analysis:

- Identifying significant terms that might indicate suicidal ideation or other mental health issues.[18]

Handling Special Cases:

- Filter non-Arabic content, pair questions, and answers, and add start and end tokens where necessary.[24]
- Also splitting the text into lines and pairs of sentences, and filtering based on length and content where the data and the purposes need that.[26]

3.1.3 Feature Extraction

After preprocessing the data extract features from it, the preprocessing and feature extraction methods ensure that the data is cleaned, transformed, and suitable for training machine learning models, improving the chatbot's accuracy and performance.[22]
Feature extraction is the process of converting raw preprocessed data into numerical features while preserving the information in the original data set. Using it presents better results than applying machine learning algorithms directly to the raw data.
Various feature extraction algorithms and techniques can be implemented for this, like:

Feature Extraction techniques

- **Bag of Words (BoW)**:
    - **Concept**: BoW represents text as a collection of words, ignoring grammar and word order. It counts the frequency of each word in a document and builds a vocabulary from all the unique words in the corpus.
    - **Use**: By focusing on word frequencies, BoW allows you to identify common and redundant elements. It's effective in simplifying text data for models like chatbots, although it doesn't capture semantic meaning or context and can't handle Ambiguity.[16]
- **TF-IDF (Term Frequency-Inverse Document Frequency)**:
    - **Concept**: TF-IDF is a numerical statistic that reflects how important a word is in a document relative to the entire corpus. It combines the frequency of a word in a document (TF) with the inverse document frequency (IDF), which downweights words that are common across many documents.
    - **Use**: TF-IDF is commonly used in text classification and sentiment analysis. It highlights words that carry significant meaning in a given context, making it ideal for applications like chatbot development and text analysis.
    - **Python Libraries**: NLTK is useful for text preprocessing (tokenization, stemming, etc.) before applying TF-IDF.[30]
- **Word Embeddings**:
    - **Concept**: Word embeddings, like those from AraVec for Arabic text, capture the semantic meaning of words by representing them as dense numerical vectors. These embeddings place semantically similar words closer in vector space.
    - **Use**: Word embeddings are especially useful for capturing the context and relationships between words. They are often pre-trained on large datasets and fine-tuned for specific tasks like chatbots.

  - **AraVec**: This is specifically designed for Arabic text and is based on the Word2Vec skip-gram model and is trained on a large corpus of Arabic text. It's a valuable tool when working with Arabic chatbots or NLP tasks requiring semantic understanding.[27]
- **N-Grams**:
  - **Concept**: N-Grams capture sequences of words (e.g., bigrams, trigrams) instead of treating words as independent entities. This helps retain word order and context.
  - **Use**: By capturing word sequences, N-Grams enhance context sensitivity in feature extraction. It is particularly useful in chatbots to detect common phrases or patterns.[25]
- **Vectorization & Featurization**:
  - **Concept**: This involves converting text into numerical vectors. Techniques include count vectorization (based on word frequency) and more sophisticated models like Conditional Random Fields (CRF) for sequence prediction.
  - **Use**: Vectorization turns raw text into features that machine learning algorithms can interpret. Count Vectorizer is simple but can miss semantic meaning or handle rare or out-of-vocabulary words that may exist. More advanced methods like supervised embeddings may be required for deeper understanding.[25][29]

NLU Techniques

- **Intent Classification**:
  - **Concept**: This identifies the user's intent or goal in a conversation. Machine learning models use the features extracted from the text to classify user queries into predefined categories (e.g., booking a flight, or asking about the weather).
  - **Use**: Intent classification is fundamental for chatbot systems as it helps determine the appropriate response based on user input.
- **Entity Extraction**:
  - **Concept**: This involves identifying specific entities in the text, such as names, dates, or product names, using techniques like part-of-speech tagging or Named Entity Recognition (NER).
  - **Use**: Entity extraction allows chatbots to pinpoint important pieces of information in a query, such as the user's location or request details, making the response more accurate and personalized.[28]

Overall, feature extraction is a process used in machine learning, data analysis, and pattern recognition to transform raw data into a set of measurable, meaningful features that can be used for further analysis or modeling. It involves identifying key attributes or characteristics from the data that are relevant to the task at hand, reducing the complexity of the data while preserving the most important information. We apply those features to make the algorithm better and some of those techniques are necessary. Many machine learning algorithms require input data in a structured form. Feature extraction transforms unstructured data (such as text, images, or signals) into a usable format for algorithms like support vector machines or neural networks. It can help us improve the model's Performance by selecting the most relevant features, models can better focus on important patterns, leading to improved accuracy, generalization, and faster training times. Also, extracted features are often more understandable and interpretable compared to raw data, allowing analysts and modelers to gain insights into the relationships and patterns within the data. In the end, these techniques above are essential for developing robust NLP models, particularly for chatbots that need to understand and respond to user input in a meaningful way.

3.1.4 Choosing the right algorithm and Training with AI algorithms

After organizing and preparing the data, the next crucial step is selecting the right algorithm for training and building the model. This decision depends on factors such as the **problem type**, **data characteristics**, and **performance requirements**. Considerations include the depth, width, and connectivity of the network architecture, which can significantly impact the model's ability to learn from data and make predictions.[16][17]

- **Algorithm Selection Criteria**
  Choosing the best algorithm depends on several factors:
  - **Task-specific goals**: Different tasks (e.g., intent classification, entity extraction, response generation) may require different algorithms.
  - **Data availability**: The size and quality of the dataset can influence the choice of model. Some algorithms perform better with large datasets, while others may work well with limited data.
  - **Computational resources**: More complex models like deep learning networks may require significant computational power, whereas simpler models can run on modest hardware.
  - **Interpretability**: Some algorithms (e.g., decision trees) are more interpretable, making them easier to understand and explain, while others (e.g., deep neural networks) offer less transparency but higher performance.
- **Experimentation and Comparison**
  Each algorithm has its strengths and weaknesses, and the choice of algorithm depends on several factors. Researchers often **experiment with multiple algorithms** and architectures to find the most suitable approach for their chatbot applications.[20] Comparison of different algorithms based on **accuracy and performance metrics** allows for an evidence-based selection. By running experiments, it's possible to identify the model that delivers the best performance on the task at hand.

- **Training and Optimization**
  Once an algorithm is selected, the model is trained on the dataset, typically divided into **training** and **test sets**. During training, the model learns to identify patterns and relationships between input features and target outputs.[22] Training involves **optimizing model parameters** to minimize prediction errors and improve performance.[33]
- **Model Evaluation**
  It's important to evaluate the model using performance metrics such as **accuracy**, **precision**, **recall**, and **F1-score**. This helps in comparing different models and algorithms based on their effectiveness in solving the problem. Comparison with Other Models by Accuracy and Performance Metrics helps us also to choose the suitable AI algorithm for our data if our data, we can test different algorithms on our data and choose the best in our model.[18]
- **Advanced AI Models and GPT-4 Performance**
  Research comparing different versions of ChatGPT, such as **ChatGPT 4.0** versus **ChatGPT 3.5**, highlights the advancements in AI models. ChatGPT 4.0 trained on a trillion parameters, showing a significant improvement in handling complex cases, such as ophthalmic challenges.[17] This indicates the importance of selecting algorithms that scale well with model size and complexity, particularly for demanding tasks.

In conclusion, selecting and training the right algorithm is a multi-step process that involves considering the nature of the problem, testing various models, and optimizing the final model based on performance metrics.[18][33]

3.1.5 Hyperparameter and parameters tuning

Hyperparameters and parameter tuning play a crucial role in optimizing the performance of chatbot models and preventing overfitting. This process involves fine-tuning several aspects, including learning rate, batch size, and regularization strength, and selecting appropriate activation functions, loss functions, optimization algorithms, and regularization techniques.

- **Activation Functions** are essential for introducing non-linearities into the neural network, enabling it to learn complex relationships. Common activation functions include Sigmoid, Tanh, ReLU (Rectified Linear Unit), and Softmax. The choice of activation function can significantly impact the model's performance, as it affects how well the model captures patterns in the data.[23]
- **Loss Functions** quantify the difference between predicted and actual values during training. Popular loss functions include Mean Squared Error (MSE) for regression tasks and Cross-Entropy Loss for classification tasks. The selection of a loss function is problem-dependent and is essential for guiding the model's learning process.[23]
- **Optimization Algorithms**, such as Stochastic Gradient Descent (SGD) and Adam, adjust the model parameters during training to minimize the loss function. While Adam is generally more efficient, some studies have shown that SGD can provide better generalization and stability, especially in certain chatbot applications.[21]
- **Regularization Techniques** like Dropout, L1/L2 Regularization, and Batch Normalization are applied to prevent overfitting, which occurs when a model performs well on training data but poorly on unseen data. Dropout, for example, randomly eliminates neurons during training to reduce reliance on specific nodes, enhancing model robustness. In some cases, L2 regularization, also known as Ridge Regression, is used to penalize large model coefficients, promoting generalization.[21]
- Tuning **hyperparameters** like the learning rate, batch size, number of layers, neurons per layer, and dropout rate are critical for optimizing performance. Grid search or random searches are commonly used methods to explore different combinations of hyperparameters. For instance, researchers have found that smaller batch sizes (e.g., 8 or 16) can yield better model accuracy, while a dropout rate of 20% to 60% is effective in preventing overfitting.[27]
- **Epochs and Early Stopping** help control the training duration and prevent the model from overfitting by monitoring performance on validation data. Training typically occurs over multiple epochs, with early stopping to halt training when performance on validation data ceases to improve. For example, a model may train for 100 epochs but stop at 22 when early stopping is triggered.[21][25]

The papers show that tuning hyperparameters like batch size, number of hidden layers, and optimizer choice significantly improves model performance. For instance, models with five hidden layers and 128 neurons per layer, combined with the Adam optimizer and a batch size of 32, achieved high accuracy in tasks like intent classification and emotion detection.
Ultimately, hyperparameter tuning, along with regularization and optimization techniques, ensures that the model can generalize well to unseen data, enhancing its ability to handle real-world conversations.[34]

3.1.6 Evaluation and Testing

After training a chatbot model, it undergoes evaluation and testing to assess its performance using various metrics like accuracy, coherence, efficiency, and user satisfaction. The chatbot is tested with sample conversations to ensure it can effectively interact with users and provide appropriate responses. Performance is typically measured through both quantitative metrics and qualitative assessments.[16][17]
One of the most commonly used evaluation metrics is accuracy, which is calculated by comparing the chatbot's performance with that of human experts or established standards. Accuracy reflects the proportion of correct predictions out of the total number of predictions. Other important evaluation metrics include[24]:

- **Precision**: The ratio of true positive predictions to all positive predictions, assessing the accuracy of the chatbot when it predicts a specific intent or entity.
- **Recall** (or Sensitivity): The ratio of true positive predictions to all actual positive instances, indicating how well the chatbot captures all relevant instances.
- **F1-score**: The harmonic mean of precision and recall, providing a balanced measure between the two metrics.
- **Confusion Matrix**: A table showing the counts of true positives, true negatives, false positives, and false negatives to visualize classification performance.
- **Loss**: Measures the error between predicted and actual values, with the type of loss function depending on the task (e.g., cross-entropy for classification).
- **Perplexity**: Used primarily for language models, it measures how well the model predicts a sample. Lower perplexity indicates a better model.[33][34]

In addition, performance metrics such as AUC (Area Under the Curve), Hamming Loss (for multi-label classification), and Kappa Statistics (Cohen's kappa for agreement between raters) provide further insights into model effectiveness.[15] The BLEU Score is used for evaluating the quality of machine-translated text by comparing it with reference translations.[21][24]

- **Area under the Curve (AUC)**: Measures the performance of binary classifiers based on the ROC curve. Higher AUC values indicate better classification ability; they range from 0 to 1.
- **Hamming Loss**: Evaluates multi-label classification models by measuring the proportion of incorrect labels. Lower values indicate better performance, with 0 being perfect.
- **Kappa Statistics (Cohen's Kappa)**: Assesses the agreement between two raters or classifiers, adjusting for chance agreement. Values range from -1 to 1, with higher values indicating stronger agreement.
- **BLEU Score**: Evaluates the quality of machine-translated text by comparing it to reference translations, based on n-gram precision. Higher scores indicate better translation accuracy.

Evaluation of chatbots in real-world scenarios often involves using test and validation datasets to ensure the model generalizes well to unseen data. This process includes monitoring metrics like validation accuracy, F1-score, and loss to prevent overfitting during training.[27][29]

The performance of chatbot systems is also assessed based on user feedback, response time, and real-world effectiveness, making both quantitative metrics (e.g., recall, confusion matrix) and qualitative assessments important for comprehensive evaluation.[19]

3.1.7 Integration and Deploying the Trained Model into Production

Once a model is trained, it needs to be deployed into a **production environment** where it can handle real-world applications. The deployment process ensures that the model is scalable, reliable, and efficient. After deployment, continuous **performance monitoring** is essential to maintain the model's effectiveness over time.[33][34]

Different Chatbot Deployment and Integration examples and components from different papers

- **University Website Integration**:
  - The chatbot in this paper is integrated into the university website, providing users with a user interface (UI) for interacting with the system.
  - **User Interaction**: The UI allows users to input queries either as text or voice, providing flexibility in interaction.
  - **Backend Processing**: The backend processes user inputs, extracts **intents**, and searches the knowledge base for relevant responses.[19]
- **Dialogflow Integration**:
  - **Dialogflow** is a natural language understanding (NLU) platform developed by Google that allows developers to create conversational interfaces, such as chatbots and voice assistants. It enables the processing of user input, extracting meaning, and managing the flow of conversation. Dialogflow can handle both text and voice input, making it suitable for integration into various applications, including websites, mobile apps, and voice-controlled devices.
  - The system here uses **Dialogflow**, a natural language understanding platform, to handle conversational interactions.
  - **Natural Language Processing (NLP)**: Dialogflow is responsible for interpreting user queries and managing the conversation flow, enhancing the chatbot's ability to understand and respond.[19]
- **Kommunicate Integration**:
  - **Kommunicate** is a customer support and chatbot integration platform designed to help businesses provide automated and real-time communication with users. It enables seamless interaction between users and chatbots on websites, mobile apps, or messaging platforms. Kommunicate allows businesses to integrate AI chatbots, like those built with Dialogflow, with live human support agents to create a smooth transition between automated and human assistance when needed.
  - **Kommunicate**, a chat plugin, is integrated into the website to add chatbot functionality, facilitating seamless communication between users and the chatbot.[19]
- **Facebook Messenger Integration**:
  - **Facebook Messenger** is a messaging app and platform developed by Meta (formerly Facebook) that allows users to send text messages, voice notes, images, videos, and even make voice or video calls. Initially launched as a simple messaging feature within Facebook, Messenger has grown into a standalone app with a wide range of functionalities, including bot

integration for businesses. It can integrate **chatbots** into Messenger to automate conversations, provide customer service, and answer user queries. Platforms like **Dialogflow** can be used to create chatbots for Messenger. It also offers APIs to enable developers to integrate chatbots or messaging functionalities into websites and apps.
    - The chatbot is also deployed here on **Facebook Messenger** to enable user interactions on that platform.
    - **Heroku Server Deployment**: To ensure 24-hour accessibility, the chatbot is hosted on a **Heroku server**, providing a reliable platform for continuous operation.[23]
- **User Interaction Interface**:
    - An interface for users to interact with the chatbot is developed using the **Angular framework**.
    - The **REST API** created on the **Django framework** is responsible for generating responses from the chatbot.
    - A **REST API** (Representational State Transfer Application Programming Interface) is a standardized approach to building APIs that allow web services to communicate using HTTP requests. REST APIs are widely used for building web services that can interact with various platforms or systems.
    - **Angular** is a popular open-source web application framework developed by Google. It is widely used for building dynamic, single-page applications (SPAs) where the content can be loaded without refreshing the entire webpage.[26]
- **Flask Framework for Admin Interface**:
    - **Flask** is a lightweight, open-source web framework for Python. It is known for being simple and flexible, making it a popular choice for developers who want to build web applications with minimal setup and configuration.
    - A separate user interface, created using the **Flask framework**, allows administrators to manage the chatbot's database.
    - This interface enables the uploading and updating of information like exam timetables and course-related details, ensuring that the chatbot's knowledge base is current and relevant.[28]

There are also other ways to deploy our effective chatbot into production to make the user use it, and there are also platforms to build and integrate our chatbot and each platform offers different levels of customization and complexity depending on the use case and development expertise. Like:

- **For Arabic and English support**, platforms like **Dialogflow**, **Rasa**, **Microsoft Bot Framework**, **IBM Watson**, and **SnatchBot** are ideal due to their advanced multilingual capabilities.
- **Dialogflow** and **Wit.ai** are great for integration with major platforms like **Facebook Messenger**.
- **Rasa** and **Microsoft Bot Framework** offer more flexibility for developers looking for highly customizable chatbots.

There are also many APIs we can build that can be utilized for seamless integration, deployment, and scaling. And frameworks like Flask and Django for deploying chatbots or web applications,

In the end, deploying and integrating a chatbot into real-world environments involves creating user interfaces, connecting with natural language processing platforms like Dialogflow, and deploying the model on scalable servers like Heroku. Different frameworks (e.g., Flask, Angular, Django) are used to manage the backend, allowing for smooth user interaction and ongoing system maintenance.[16][20][23][25]

Successfully integrating and deploying a chatbot involves several critical steps, ensuring the system is functional, scalable, and user-friendly. Here's a comprehensive guide to the right steps from the discussed technologies and platforms:

1. **Prepare the Chatbot model:**
    - **Model Preparation**: After the previous steps end we prepare our effective powerful model for the integration with user by deploying in web, mobile, or desktop application.
2. **Develop the User Interface:**
    - **Front-End**: Build a dynamic UI using frameworks like Angular, supporting text and voice inputs.[23]
    - **Admin Interface**: Create an admin panel using Flask to manage and update the chatbot's data.[25]
3. **Set Up Backend Processing:**
    - **Backend**: like using Django or Flask to create REST APIs that handle requests and process user inputs.[23][25]
    - **NLP Integration**: Integrate NLP platforms like Dialogflow for understanding and managing conversations.[16]
4. **Deploy the Chatbot:**
    - **Hosting**: Deploy on scalable servers like Heroku for reliable, continuous operation.[20][23]
    - **Platform Integration**: Integrate with messaging platforms like Facebook Messenger for broader reach.[20]
5. **Monitor and Maintain:**
    - **Performance**: Continuously track and improve the chatbot's performance based on user interactions and feedback.
    - **Updates**: Regularly update the chatbot's knowledge base and algorithms to keep it effective.
6. **Customization and Multilingual Support:**
    - **Customization**: Choose platforms like Rasa or Microsoft Bot Framework for advanced customization.
    - **Multilingual Support**: Ensure the platform supports multiple languages if needed.

By following these steps, you can effectively integrate and deploy a chatbot, ensuring it delivers a seamless and interactive experience for users. Leveraging the right technologies and platforms will enable you to create a robust and scalable chatbot solution tailored to your needs.

3.2 Used Machine and Deep Learning

In building intelligent chatbot agents we can use different machine and deep learning algorithms for training and building processes on our collected data.

Machine learning (ML) and Deep learning (DL) are subsets of AI, DL are subset of ML used to enable systems to learn from data and improve their performance over time without being explicitly programmed and effectively capture complex patterns and relationships in the data using DL[33][34]. Machine and deep learning play a crucial role in building intelligent chatbots by enhancing their ability to understand and interact with users. They enable chatbots to classify user intents, extract relevant entities like names or dates, and respond appropriately based on the context of the conversation. Models like Support Vector Machines (SVM) and Random Forests help with intent classification, while deep learning architectures such as RNNs, LSTMs, and Transformers excel at natural language understanding (NLU) and generating human-like responses. Additionally, machine learning aids in personalizing conversations and maintaining coherence across longer interactions. With the help of neural machine translation (NMT), chatbots can even handle multilingual conversations efficiently. These advancements allow chatbots to become more accurate, adaptive, and capable of handling complex queries in real-time.

Through the papers we compared to see which algorithms were the most used in training the chatbot model as we see in Table 3.

Table 3. Used Deep Learning algorithms and papers

ML, DL algorithms	Paper resources	overview
Support Vector Machine (SVM)	[23][25][27][30]	**Purpose**: A supervised learning algorithm used for classification and regression tasks. The basic idea of SVM is to find a hyperplane that best separates the classes in the feature space. **Key Concepts**: o **Hyperplane**: A line (in 2D) or a plane (in higher dimensions) that separates classes. o **Support Vectors**: Data points closest to the hyperplane, influencing its position. o **Kernel Trick**: Transforms input space into a higher-dimensional space to handle non-linear data. o **Margin**: Maximizes the distance between the hyperplane and the nearest data points from each class. o **Benefits**: Effective in high-dimensional spaces and robust to overfitting, especially in situations where the number of dimensions exceeds the number of samples.
Support Vector Classifier (SVC)	[18][21]	**Purpose**: A type of SVM that is used specifically for classification tasks.
K-Nearest Neighbors (KNN)	[18]	**Purpose**: Classifies or predicts, and a non-parametric, lazy learning algorithm used for both classification and regression tasks and based on the k-nearest data points in the feature space and assigns a label based on the majority class among its neighbors (for classification) or averages the labels (for regression) using distance metrics like Euclidean and Manhattan distances. **Key Concepts**: o **K**: Number of nearest neighbors to consider. o **Distance Metrics**: Measures similarity between data points (e.g., Euclidean, Manhattan). o **Voting**: For classification, the majority class among neighbors is chosen; for regression, the average of neighbors' values is used.
Multinomial Naive Bayes (MNB)	[18][26][30]	**Purpose**: probabilistic machine learning algorithm based on Bayes' theorem, with an assumption of independence between features. Despite its simplicity, it often performs surprisingly well in classification tasks, particularly when the independence assumption holds true or is not very detrimental. **Key Concepts**: o **Bayes' Theorem**: Calculates the probability of an event based on prior knowledge. o **Independence Assumption**: Features are assumed to be independent of each other. o **Likelihood and Prior**: Computes the likelihood of each class given the feature values and the prior probability of each class.
Logistic Regression (LR)	[18][19]	**Purpose**: A binary classification algorithm using the sigmoid function to map outputs between 0 and 1. In spite of its name, it is not utilized for regression but for classification. **Key Concepts**: o **Sigmoid Function**: Maps the output to a value between 0 and 1, representing the probability of belonging to the positive class. o **Decision Boundary**: Separates classes based on predicted probabilities (probability > 0.5 = positive class). o **Cost Function**: Uses binary cross-entropy (log loss) to measure the

		difference between predicted probabilities and actual labels, minimizing this cost during training.
Random Forest (RF)	[18]	**Purpose**: An ensemble learning method building multiple decision trees for classification or regression. **Key Features**: ○ **Ensemble Method**: Creates multiple decision trees, each trained on random subsets of the data and features, and the final prediction is made by aggregating the predictions of individual trees (e.g., by averaging for regression or using voting for classification). ○ **Aggregation**: Combines predictions from individual trees (e.g., voting for classification, averaging for regression). ○ **Benefits**: Robust against overfitting, handles high-dimensional data well, and is interpretable.
Convolutional Neural Networks (CNNs)	[17][20][21][22][23]	**Purpose**: Process grid-like data such as images, using convolution and pooling layers. **Key Concepts**: ○ **Convolutional Layers**: Detect patterns and features. ○ **Pooling Layers**: Downsample feature maps. ○ **Application in NLP**: Useful for tasks like sentiment analysis or intent classification.
Recurrent Neural Networks (RNNs) and like: LSTM, GRU	[18][20][21][22][23] [24][25][26][27][28]	**Purpose**: Sequential data processing, used for tasks like text analysis, time series prediction, speech recognition, and dialogue management. **Key Concepts**: ○ **Recurrent Connections**: Allow information to persist across sequences. ○ **Hidden State**: Represents memory, updated at each time step based on the input and previous state. ○ **Backpropagation Through Time (BPTT)**: A variant of backpropagation used to train RNNs.
Long Short-Term Memory (LSTM)	[18][21][22][23][24] [25][26][27][28]	**Purpose**: Address vanishing gradient problems and capture long-term dependencies in sequential data. **Key Concepts**: ○ **Memory Cells**: Maintain information over long sequences. ○ **Gates**: Input, forget, and output gates control the flow of information. ○ **Cell State**: Carries relevant information through sequences.
Gated Recurrent Unit (GRU)	[18][21][22][23][24] [25][25][27]	**Purpose**: Simplified version of LSTM, retaining the ability to capture long-range dependencies. **Key Concepts**: ○ **Update Gate**: Controls memory retention. ○ **Reset Gate**: Decides how much of the past information to forget.
Seq2Seq model with specific attention to recurrent neural network (RNN) architectures such as Long Short Term Memory (LSTM) or Gated Recurrent Unit (GRU)	[16][20][22][21][23]	**Purpose**: Transform input sequences to output sequences, effective for tasks like machine translation and text summarization. **Architecture**: ○ **Encoder**: Compresses input sequence into a context vector. ○ **Decoder**: Generates output sequence from the context vector. ○ **Key Models**: Use RNNs (LSTMs, GRUs) or **Transformers** (more recent, using self-attention mechanisms).[
Enhancement of RNN-GRU	[2]	**Purpose**: This technique is improved by adding three additional cells which are Refinement, Adjustment, and Output cells.
Hybrid (GRU-LSTM)	[24]	**Purpose**: Combines the GRU and LSTM architectures to leverage the strengths of both models. Uses GRU as the encoder and LSTM as the decoder in a sequence-to-sequence framework. Aims to balance between the simplicity and efficiency of GRU with the long-term dependency handling capabilities of LSTM
Bidirectional Recurrent Neural Network (BiRNN)	[22]	**Purpose**: Process sequences in both forward and backward directions, improving performance by capturing past and future context. **Variants**: BiLSTM, BiGRU.
Bidirectional Long Short-Term Memory (BiLSTM)	[18][27]	**Purpose**: Bidirectional version of LSTM for enhanced sequence modeling and to model sequential data, such as text. It's widely used in tasks like text classification, language modeling, and named entity recognition (NER).
Bidirectional Gated Recurrent Unit (BiGRU)	[18][27]	**Purpose**: Bidirectional GRU, that processes input sequences in both forward and backward directions. It captures dependencies from past and future contexts simultaneously, enabling a better understanding of the sequence data and it is commonly used in tasks such as natural language processing (NLP), where the meaning of a word can depend on both preceding and succeeding words.
Convolutional Long Short-Term Memory (CLSTM)	[18]	**Purpose**: Combine LSTM's temporal modeling with convolutional layers, useful for tasks requiring both spatial and temporal information (e.g., video analysis).
Attention mechanisms besides BiGRU	[2]	**Purpose**: Attention mechanisms Improved technique of seq2seq learning based on RNNs cell. Fix the issue of systems' inability to retain longer sequences.
Deep Belief Network (DBN)	[19]	**Purpose**: Generative models using multiple layers of stochastic, latent variables

		for unsupervised learning. **Key Concepts**: ○ **Restricted Boltzmann Machines (RBMs)**: The building blocks of DBNs. ○ **Unsupervised Pretraining**: Prepares the network by learning features in a layer-wise manner.
Stochastic Gradient Descent (SGD)	[19]	**Purpose**: Optimization algorithm used for training neural networks. **Variants**: Adam, which improves convergence speed and stability.
Deep Reinforcement Learning (DRL)	[24]	**Purpose**: Train agents to make decisions through trial and error, optimizing cumulative rewards. **Application**: this can be used in chatbots to optimize response generation based on user interactions.
Transformer models	[17][20][22][25][28]	**Purpose**: Process sequences in parallel using self-attention mechanisms, excelling in translation and text generation. **Key Architectures**: ○ **BERT (Bidirectional Encoder Representations from Transformers)**: Specialized for NLP tasks. ○ **GPT (Generative Pre-trained Transformer)**: Widely used in generative tasks. ○ **Key Components**: Encoder and decoder layers with multi-head self-attention.
Dual Intent And Entity Transformer (DIET)	[28][29]	**Purpose**: A lightweight architecture for intent classification and entity recognition, commonly used in chatbots. **Key Features**: ○ **Dual Task**: Simultaneously handles intent classification and entity extraction. ○ **Transformer-based**: Incorporates self-attention mechanisms for language understanding. ○ **Pretrained Embeddings**: Uses BERT, GloVe, etc., for better performance.
Neural Machine Translation (NMT)	[22]	**Purpose**: NMT is a type of machine translation that uses neural networks to predict the likelihood of a sequence of words, typically translating sentences from one language to another. NMT models, like seq2seq models with attention mechanisms, have significantly improved translation quality compared to traditional methods

In the end, these methods collectively offer powerful tools for handling various tasks such as classification, regression, sequential data analysis, and more complex neural architectures for specific applications like here in our intelligent assistants Chatbots.

3.3 Strength and Limitations in algorithms

Each algorithm has its strengths and weaknesses, and the choice of algorithm depends on factors such as the specific task, available data, computational resources, and desired level of interpretability. Researchers typically experiment with multiple algorithms and architectures to find the most suitable approach for their chatbot application.[20]
The effectiveness of each algorithm may depend on factors such as the complexity of the dataset, the nature of user queries, and the specific requirements of the application. The choice of algorithm may involve trade-offs between accuracy, computational efficiency, and scalability. Limitations may also arise from the quality and quantity of the training data, as well as the preprocessing techniques applied to the data.[21] That is why we will here talk about some algorithms that are used in chatbot model-building strengths and limitations, and take part in chatbot accuracy and efficiency.

In Machine Learning:

1. RF (Random Forest)

- **(Limitations)**
  - Random Forest models can become computationally expensive and slow with a large number of trees or very large datasets. They also tend to overfit if not properly regularized and may not perform as well on high-dimensional data compared to deep learning models.[18]
- **(Strength)**
  - Random Forests can be used for both classification and regression tasks.[18]
  - Less prone to overfitting compared to individual decision trees, especially with proper tuning.[18]
  - It can provide insights into feature importance, helping with feature selection.[18]
  - Can handle complex relationships and interactions in data without needing feature scaling.[18]

2. SVM (Support Vector Machine)

- **(Limitations)**
  - Computational Complexity: SVM can be computationally intensive, especially when dealing with large datasets or high-dimensional feature spaces. This can result in longer training times and increased resource requirements.[30]

- Sensitivity to Parameter Tuning: SVM performance can be sensitive to the selection of hyperparameters such as the choice of kernel function and regularization parameter.[30]
- Suboptimal parameter settings may lead to reduced classification accuracy.[30]
- Difficulty with Large Datasets: SVM may struggle with scalability when applied to very large datasets, as training time and memory requirements can become prohibitive.[30]
- Inefficiency with Imbalanced Data: SVM may not perform well when dealing with imbalanced datasets, where one class significantly outnumbers the others. In such cases, the classifier may exhibit bias towards the majority class.[30]

- **(Strength)**
  - SVM performs well when there is a clear margin of separation between classes.[30]
  - SVM can handle nonlinear classification using kernel functions like radial basis function (RBF).[30]
  - Particularly effective in cases where the number of dimensions exceeds the number of samples.[30]

3. SVC (Support Vector Classifier)

- **(Limitations)**
  - SVCs can be computationally intensive, especially with large datasets, as they require significant memory and processing power. The choice of the kernel and hyperparameters greatly affects performance, and SVCs do not scale well to large datasets.[18]
- **(Strength)**
  - Effective for Small to Medium Datasets: Performs well on moderately sized datasets with clean data.[18]
  - High Accuracy: Can achieve high classification accuracy when properly tuned, especially for binary classification.[18]

4. LR (Logistic Regression)

- **(Limitations)**
  - Logistic Regression is relatively simple and may not capture complex patterns in data as effectively as more sophisticated models. It also assumes a linear relationship between the input features and the log-odds of the target, which might not hold true in many real-world scenarios.[18]
  - Assumes a linear relationship between the independent variables and the log-odds of the dependent variable, which may not always hold true.[19]
  - Limited ability to capture complex nonlinear relationships in the data.[19]
  - vulnerable to overfitting if the model is too complex relative to the size of the dataset.[19]
- **(Strength)**
  - Easy to implement and interpret, especially for binary classification tasks.[18][19]
  - Fast to train and evaluate, particularly for smaller datasets.[18][19]
  - Requires minimal computational resources compared to more complex models.[18][19]

5. MNB (Multinomial Naive Bayes)

- **(Limitations)**
  - Naive Bayes models assume feature independence, which is often not true in practice, leading to poorer performance on datasets with correlated features. They are also not suitable for capturing complex patterns and interactions between features.[18]
  - Assumption of Independence: Naïve Bayes relies on the strong assumption of feature independence, which may not hold true for all datasets. In reality, features in text data (such as words in tweets) may exhibit dependencies that violate the independence assumption.[30]
  - Sensitivity to Feature Correlation: Naïve Bayes may struggle to capture relationships between correlated features, leading to suboptimal performance in datasets where feature dependencies are present.[30]
  - Handling of Out-of-Vocabulary Words: Naïve Bayes may struggle to handle out-of-vocabulary words (i.e., words not seen during training) effectively. This can result in difficulties generalizing to new or unseen data.[30]
  - Limited Expressiveness: Naïve Bayes has limited expressiveness compared to more complex models. While its simplicity can be advantageous for certain tasks, it may also result in suboptimal performance when dealing with complex relationships in the data.[30]
- **(Strength)**
  - Very fast to train and classify, especially for text classification tasks like spam detection. [18][30]
  - Easy to implement and interpret due to its simple probabilistic foundation. [18][30]
  - Can handle small datasets well, assuming the feature independence assumption holds. [18][30]

6. K-Nearest Neighbors (KNN)

- **(Limitations)**
  - Computationally expensive for large datasets as it requires calculating distances between each data point.[19]
  - Sensitive to irrelevant features and noise in the data.[19]
  - Need to determine the optimal value of K, which can impact the model's performance.[19]
- **(Strength)**

- Easy to implement and understand.[19]
- KNN is a lazy learning algorithm that doesn't require a training phase, making it simple for small datasets.[19]
- Can be used for both classification and regression tasks. [19]

In Deep Learning:

1. Sequence-to-Sequence (Seq2Seq) Modeling:

- **(Limitations)**
  - Seq2Seq models may suffer from issues like generating generic or uninformative responses, known as "hallucinations."[20]
  - They may also struggle with handling rare or out-of-vocabulary words.[20]
    - **With LSTM and GRU**
  - Training Seq2Seq models with LSTM or GRU architectures can be computationally intensive and require significant computational resources.[16]
  - These models often require large amounts of training data to learn effectively, which may not always be readily available, especially for specialized domains.[16]
  - Without proper regularization techniques and hyperparameter tuning, Seq2Seq models may be prone to overfitting, resulting in poor generalization to unseen data.[16]
  - While LSTM and GRU cells can capture some long-range dependencies, they may still struggle with understanding and maintaining context over extended dialogue sequences.[16]
    - **With Attention Mechanism**
  - Can be computationally expensive, especially with large vocabulary sizes, due to the attention mechanism's need to attend to all words in the input sequence.[22]
  - Vulnerable to generating generic or repetitive responses if not properly trained or if the training data lacks diversity.[22]
  - May struggle with handling out-of-domain or contextually ambiguous inputs, leading to inaccurate or nonsensical responses in certain scenarios.[22]
- **(Strength)**
  - Seq2Seq models are effective for tasks like response generation in chatbots. They can capture semantic meaning and context in user queries and generate contextually relevant responses.[20]
  - Support variable-length size of input and response.[2]
    - **With LSTM and GRU**
  - Seq2Seq models with LSTM and GRU excel in handling sequential data, particularly in language-based tasks. LSTMs are stronger at managing long-term dependencies due to their memory cells, while GRUs offer a more computationally efficient alternative. Both architectures contribute to the versatility and effectiveness of Seq2Seq models in NLP applications.[16]
    - **With Attention Mechanism**
  - Enable variable-length input and output sequences.[22]
  - The attention mechanism allows the model to focus on relevant parts of the input sequence when generating the output sequence.[22]

2. Recurrent Neural Networks (RNNs)

- **(Limitations)**
  - Vanishing/Exploding Gradient Problem: Traditional RNNs are prone to vanishing and exploding gradient issues, particularly when dealing with long sequences. This limitation hinders their ability to learn and capture long-term dependencies effectively, leading to difficulties in generating coherent and contextually relevant responses.[20][22][23][25]
  - Limited Memory and Context Understanding: RNNs have a fixed-size hidden state, which limits their memory capacity and ability to retain information over long sequences. This can lead to challenges in capturing complex relationships and understanding long-term context in conversations.[22][23][25]
  - Computational Complexity: Training traditional RNNs can be computationally intensive, especially when handling large datasets, large vocabularies, or complex architectures. Sequential processing during training further adds to the computational burden.[22][25]
- **(Strength)**
  - RNNs are well-suited for sequential data processing tasks like dialogue management. They can capture temporal dependencies and long-term context in conversations.[20]
  - Well-suited for sequential data processing, making them suitable for modeling conversational data.[22]
  - Can capture temporal dependencies in sequences.[22]

3. Attention mechanisms besides BiGRU: and Enhancement of RNN-GRU

- **(Limitations)**
  - Suffer from gradient exploding and vanishing problems.[2]

- Difficult to process very long sequences.[2]
- Not suitable for parallelizing or stacking up.[2]

- **(Strength)**
  - Uses less training parameters.[2]
  - Uses less memory.[2]
  - Take less time in execution.[2]
  - Less complex structure.[2]

4. GRU (Gated Recurrent Unit)

- **(Limitations)**
  - Not suitable for large datasets and long-distance relations.[2]
  - GRUs, while efficient, can struggle with capturing long-term dependencies, particularly in very long sequences and complex patterns. This limitation is noted in comparison to LSTMs and more sophisticated architectures like Transformers. GRUs may face challenges in modeling long-distance relationships within data and may not perform as effectively on tasks that require extensive context awareness.[18][22][23][24][26][27]
  - Although GRUs are computationally less intensive than LSTMs, they can still require significant resources, especially when working with large datasets. Training large-scale GRU models can be computationally expensive in terms of memory and processing power.[21][23][27]
  - Like LSTMs, GRUs are susceptible to overfitting, particularly when working with small datasets or complex architectures. Careful tuning of hyperparameters and regularization techniques are necessary to mitigate this issue.[21]
  - GRUs have a simpler architecture with fewer parameters compared to LSTMs, which may result in reduced expressiveness and representational capacity. This simplicity can lead to limitations in capturing complex patterns and relationships within sequential data.[25][26][27]
- **(Strength)**
  - Can capture dependencies in sequential data effectively.[22]
  - GRU tends to train faster than LSTM and has lower memory requirements due to its simpler structure with fewer gates, making it computationally efficient.[2][22][26]
  - Take less time in execution.[2]
  - Less complex structure.[2]
  - In compression to LSTM, GRU requires fewer parameters training and it not being required for an additional cell state.[2]

5. LSTM (Long Short-Term Memory)

- **(Limitations)**
  - LSTM models are more complex and computationally intensive compared to GRUs and traditional RNNs, requiring significant memory and processing power. This complexity, combined with a more resource-heavy architecture, leads to slower training times, particularly for large datasets and complex models. LSTMs also demand more careful tuning of hyperparameters, as they are prone to overfitting, especially when the dataset is small or regularization techniques are insufficient. Managing their computational complexity and preventing overfitting are key challenges when working with LSTMs in large-scale applications. [18][21][22][23][25][26][27]
  - LSTMs are effective at capturing long-term dependencies in sequential data, but they face computational inefficiencies with larger datasets and complex architectures. Despite improvements, LSTMs may struggle with very long sequences, requiring significant resources and still occasionally missing long-range dependencies. [21][23][24][27]
  - While LSTMs improve upon traditional RNNs leading to longer training times and potentially more challenging optimization and by mitigating the vanishing gradient problem, they are still susceptible to it, especially when dealing with very long sequences. This issue can limit their ability to effectively capture long-term dependencies in sequential data. [21][22][25][26]
  - Overfitting is a significant concern for LSTM models, particularly when the dataset is small or the architecture is overly complex. Careful tuning of hyperparameters and regularization techniques are necessary to prevent overfitting and ensure generalization. [22][23][28]
  - Difficulty in parallelization during training can affect efficiency on hardware with limited resources.[23]
  - LSTMs require significant memory resources, especially as the size of the network and the length of sequences increase, which can limit their scalability in resource-constrained environments.[28]
  - While LSTMs are capable of capturing sequential information, they may struggle with capturing complex contextual dependencies, especially in conversational contexts where understanding the broader context is crucial for accurate responses.[28]
- **(Strength)**
  - Address the vanishing/exploding gradient problem in RNNs by introducing memory cells.[22]
  - Capturing Long-Term Dependencies: LSTM is specifically designed to address the vanishing gradient problem and is capable of capturing long-term dependencies in sequential data.[22][26]

- LSTM has a more complex architecture with additional gates, allowing it to capture more complex patterns in the data.[26]

6. BiLSTM (Bidirectional Long Short-Term Memory)

- **(Limitations)**
  - BiLSTMs can be computationally expensive and require more memory due to their bidirectional nature. They can also be more challenging to train and may suffer from vanishing or exploding gradients in very deep networks.[18]
  - Bi-directional architectures can enhance the model's ability to capture contextual information from both past and future inputs. However, they also introduce more parameters, leading to increased computational complexity and potentially higher risk of overfitting.[27]
  - Despite their effectiveness, bi-directional models may require more training data to generalize well, especially in cases where the datasets are limited.[27]
- **(Strength)**
  - BiLSTM models can capture information from both past and future inputs, which enhances their ability to understand the broader context of the sequence. [18][27]
  - They are effective in handling tasks where context from both directions is important, such as language modeling and speech recognition. [18][27]

7. BiGRU (Bidirectional GRU)

- **(Limitations)**
  - Similar to BiLSTM, Bidirectional GRUs can be computationally expensive and require more memory. They may also face challenges in training and tuning and can suffer from overfitting if not properly regularized.[18]
  - Bi-directional architectures can enhance the model's ability to capture contextual information from both past and future inputs. However, they also introduce more parameters, leading to increased computational complexity and potentially higher risk of overfitting.[27]
  - Despite their effectiveness, bi-directional models may require more training data to generalize well, especially in cases where the datasets are limited.[27]
- **(Strength)**
  - Like BiLSTM, BiGRU can capture contextual information from both past and future sequences, making it more accurate in tasks involving sequential data. [18][27]
  - GRUs are more computationally efficient than LSTMs, and using a bidirectional architecture improves their ability to handle longer sequences. [18][27]

8. Convolutional Neural Networks (CNNs)

- **(Limitations)**
  - CNNs may struggle with capturing long-range dependencies in sequential data, which can be important for dialogue management. They may also require large amounts of data for training and can be sensitive to the choice of hyperparameters.[20,23]
  - CNNs typically operate on fixed-size inputs and may struggle with handling variable-length text sequences.[21]
  - Capturing long-term dependencies in text data, which is essential for understanding the context of user queries, may be challenging for CNNs compared to recurrent neural networks (RNNs) like LSTM and GRU.[21]
  - CNNs are primarily designed for image data and may not be the most suitable choice for sequential or text data like chatbot conversations.[21,23]
  - Require significant computational resources, especially for deeper architectures.[23]
- **(Strength)**
  - CNNs are effective for tasks like natural language understanding (NLU) and sentiment analysis. They can capture spatial features in text data and are computationally efficient.[20]

9. CLSTM (Convolutional LSTM)

- **(Limitations)**
  - CLSTMs, which combine convolutional and LSTM layers, can be quite complex and require significant computational resources. They may also be harder to tune and prone to overfitting on small datasets.[18]
- **(Strength)**
  - By combining convolutional layers with LSTM, CLSTM can capture both spatial and temporal dependencies, which is valuable in tasks that involve sequential data with spatial relationships, such as video analysis.[18]

10. Deep Belief Network (DBN)

- **(Limitations)**
  - Requires a large amount of labeled training data to effectively learn complex patterns.[19]
  - Training deep networks can be computationally expensive and time-consuming.[19]
  - May suffer from the vanishing gradient problem, especially in deep architectures, leading to slow convergence during training[19]
- **(Strength)**

- DBNs can pre-train models in an unsupervised manner, making them useful for tasks with limited labeled data.[19]
- They are capable of learning complex patterns in data, making them effective in applications like image recognition and generative tasks.[19]

11. Transformer Architecture (like BERT, GPT, etc.)

- **(Limitations)**
  - Transformers can be computationally expensive and may require large amounts of data for training. They may also be less interpretable compared to traditional RNN architectures.[20]
  - Require large amounts of data and computational resources for training, making them less accessible for smaller-scale projects or applications with limited resources.[22]
  - Limited by fixed-length input sequences, which may pose challenges in handling very long or variable-length sequences efficiently.[22]
  - May struggle with out-of-vocabulary words or rare language patterns, especially if not pre-trained on a sufficiently diverse dataset.[22]
- **(Strength)**
  - They are highly parallelizable and have achieved state-of-the-art results in various natural language processing tasks.[20]
  - Capture bidirectional dependencies in text effectively.[22]
  - Transformers, with their attention mechanism, Can handle long-range dependencies without recurrence, making them more parallelizable, potentially faster, and focusing on relevant parts of the conversation.[20, 22]

12. Hybrid (GRU-LSTM) Model

- **(Limitations)**
  - While the hybrid model aims to combine the strengths of both GRU and LSTM architectures, it may also inherit some of their respective limitations. Balancing the trade-offs between simplicity, efficiency, and long-term dependency handling may pose challenges in model design and optimization.[24]
- **(Strength)**
  - A hybrid model combines the efficiency of GRU with the long-term memory capabilities of LSTM, potentially improving performance on tasks requiring both efficiency and context awareness.[24]

13. Dual Intent and Entity Transformer (DIET)

- **(Limitations)**
  - Transformer architectures, including DIET, can be computationally expensive to train, especially on large datasets, due to their self-attention mechanisms and parallel processing, leading to longer training times.[28]
  - Transformer models are susceptible to overfitting, especially when trained on relatively small datasets, which may lead to suboptimal performance on unseen data.[28]
  - Like LSTMs, transformer models require significant computational resources and memory, which can limit their deployment in resource-constrained environments, particularly on devices with limited processing power.[28]
  - Potential limitations could include scalability issues with large datasets, sensitivity to hyperparameters, or challenges in handling noisy or ambiguous input.[29]
- **(Strength)**
  - DIET models can efficiently handle intent classification and entity extraction simultaneously, making them effective for tasks like chatbot intent understanding.[28][29]
  - Transformers, including DIET, are highly scalable, allowing them to be applied to large datasets.[28][29]
  - DIET leverages the strengths of the transformer architecture, offering superior performance in tasks like intent recognition and language understanding.[28][29]

14. SGD (Stochastic Gradient Descent)

- **(Limitations)**
  - While fast and scalable, SGD can be sensitive to the choice of hyperparameters like the learning rate. It often requires extensive tuning and may struggle with convergence on complex, non-convex problems, leading to suboptimal performance.[18]
- **(Strength)**
  - Highly efficient and suitable for large-scale datasets, particularly for linear models.[18]
  - Scalable and fast, especially for online learning tasks.[18]
  - Easy to implement and requires less memory.[18]

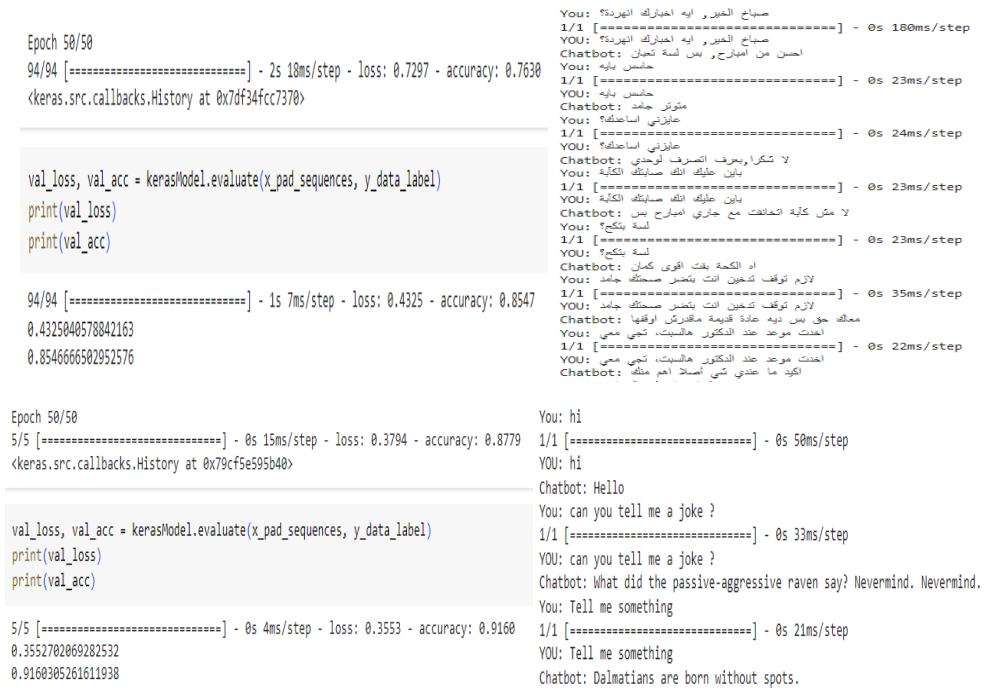

Fig 5.Arabic and English Chatbots with simple RNN algorithm

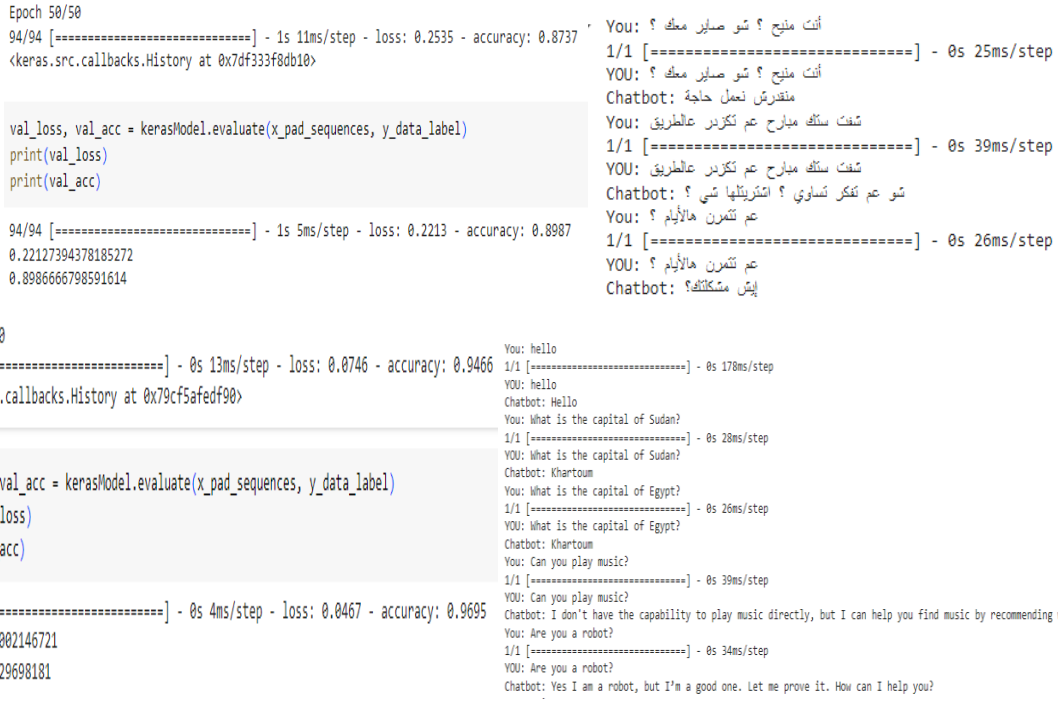

Fig 6.Arabic and English Chatbots with GRU algorithm

Fig 7.Arabic and English Chatbots with LSTM algorithm

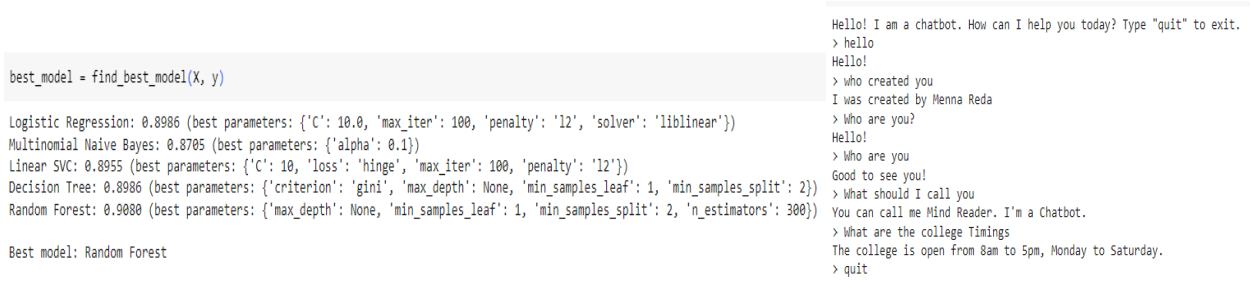

Fig 8.Arabic and English Chatbots with ML(RF) algorithm

Conclusion and Future Work

Chatbot design is a multidisciplinary field that integrates natural language processing (NLP), machine learning (ML), deep learning (DL), and artificial intelligence (AI) to enable intelligent, dynamic interactions. The evolution from rule-based systems to AI-powered chatbots has significantly improved user experience through more context-aware and flexible responses.

Despite these advancements, challenges persist, especially in understanding complex contexts, and emotional nuances, and addressing ethical concerns. Arabic chatbots, in particular, face unique hurdles due to the complexity of the Arabic language, limited research, and the predominance of rule-based systems over more advanced AI techniques. The scarcity of datasets and lack of deep learning applications for Arabic also contribute to slower progress. Also, there are more problems in Arabic chatbots for different other reasons. However, chatbots hold tremendous potential to transform industries such as healthcare, education, and customer service, offering personalized assistance and streamlining operations. The continuous evolution of chatbot technology is driven by technological advancements, user-centered design, and collaborative ecosystems.

For Arabic chatbots, addressing the limitations of language complexity, enhancing dataset availability, and incorporating dialect-specific solutions will be key to overcoming current challenges. Ethical considerations and a focus on user-centric development remain paramount as chatbots continue to revolutionize human-computer interactions across the globe.

We concluded also that current chatbots are still unable to simulate human conversation. Simultaneously, increasing research interest and rapid technological advancements could evolve chatbot conversation and make chatbots more flexible, fluent, and human-like. Indeed, this SLR provides various limitations for Arabic, and English chatbots and traditional and intelligent chatbots, which creates a chance for researchers to continue to develop research and try to overcome them on chatbots.

References

[1] Ahmed, A., Ali, N., Alzubaidi, M., Zaghouani, W., Abd-alrazaq, A., & Househ, M. (2022). Arabic chatbot technologies: A scoping review. Computer Methods and Programs in Biomedicine Update, 2, 100057.

[2] Alsheddi, A. S., & Alhenaki, L. S. (2022). English and arabic chatbots: A systematic literature review. International Journal of Advanced Computer Science and Applications, 13(8).

[3] Saoudi, Y., & Gammoudi, M. M. (2023). Trends and challenges of Arabic Chatbots: Literature review. Jordanian Journal of Computers and Information Technology (JJCIT), 9(03).

[4] Alabbas, A., & Alomar, K. (2024). Tayseer: A Novel AI-Powered Arabic Chatbot Framework for Technical and Vocational Student Helpdesk Services and Enhancing Student Interactions. Applied Sciences, 14(6), 2547.

[5] Shukla, M., Goyal, I., Gupta, B., & Sharma, J. (2024). A Comparative Study of ChatGPT, Gemini, and Perplexity. International Journal of Innovative Research in Computer Science & Technology, 12(4), 10-15.

[6] Rane, N., Choudhary, S., & Rane, J. (2024). Gemini versus ChatGPT: applications, performance, architecture, capabilities, and implementation. Performance, Architecture, Capabilities, and Implementation (February 13, 2024).

[7] Caldarini, G., Jaf, S., & McGarry, K. (2022). A literature survey of recent advances in chatbots. Information, 13(1), 41.

[8] Adamopoulou, E., & Moussiades, L. (2020). Chatbots: History, technology, and applications. Machine Learning with applications, 2, 100006.

[9] Hu, K. (2023). ChatGPT sets record for fastest-growing user base-analyst note. reuters, 12, 2023.

[10] Marous, J. (2018). Meet 11 of the most interesting chatbots in banking. The Financial Brand, 14.

[11] Grudin, J., & Jacques, R. (2019, May). Chatbots, humbots, and the quest for artificial general intelligence. In Proceedings of the 2019 CHI conference on human factors in computing systems (pp. 1-11).

[12] Abdul-Kader, S. A., & Woods, J. C. (2015). Survey on chatbot design techniques in speech conversation systems. International Journal of Advanced Computer Science and Applications, 6(7).

[13] Park, D. M., Jeong, S. S., & Seo, Y. S. (2022). Systematic review on chatbot techniques and applications. Journal of Information Processing Systems, 18(1), 26-47.

[14] Kumar, R., & Ali, M. M. (2020). A review on chatbot design and implementation techniques. Int. J. Eng. Technol, 7(11), 2791-2800.

[15] Ashfaque, M. W., Tharewal, S., Iqhbal, S., & Kayte, C. N. (2020, October). A Review on Techniques, Characteristics and approaches of an intelligent tutoring Chatbot system. In 2020 International Conference on Smart Innovations in Design, Environment, Management, Planning and Computing (ICSIDEMPC) (pp. 258-262). IEEE.

[16] Karri, S. P. R., & Kumar, B. S. (2020, January). Deep learning techniques for implementation of chatbots. In 2020 International conference on computer communication and informatics (ICCCI) (pp. 1-5). IEEE.

[17] Madadi, Y., Delsoz, M., Khouri, A. S., Boland, M., Grzybowski, A., & Yousefi, S. (2024). Applications of artificial intelligence-enabled robots and chatbots in ophthalmology: recent advances and future trends. Current Opinion in Ophthalmology, 10-1097.

[18] Elsayed, N., ElSayed, Z., & Ozer, M. (2024). CautionSuicide: A Deep Learning Based Approach for Detecting Suicidal Ideation in Real Time Chatbot Conversation. arXiv preprint arXiv:2401.01023.

[19] Dhotre, D. R., Jain, N., Mhaske, C., Choubey, N., & Patil, D. D. (2024). ADiTi App: Leveraging Deep Learning and Generative AI for a Chatbot Application with Deep Belief Networks. International Journal of Intelligent Systems and Applications in Engineering, 12(1s), 569-576.

[20] Ali, A., Abbas, T., & Ali, D. (2024). Advancing Chatbot Technology: Deep Learning and Meta-Analysis Integration.

[21] Deepa, A., Thumati, S. S., & Reyya, S. (2022, April). An efficient deep learning based chatbot for GRIET. In 2022 IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE) (pp. 1-5). IEEE.

[22] Dhyani, M., & Kumar, R. (2021). An intelligent Chatbot using deep learning with Bidirectional RNN and attention model. Materials today: proceedings, 34, 817-824.

[23] Hailu, G. Y., & Welay, S. (2024). Deep Learning Based Amharic Chatbot for FAQs in Universities. arXiv preprint arXiv:2402.01720

[24] Naveen, P., Haw, S. C., Nadthan, D., & Ramamoorthy, S. K. (2023). Improving Chatbot Performance using Hybrid Deep Learning Approach. Journal of System and Management Sciences, 13(3), 505-516.

[25] Ahajri, A., & Alzubi, R. (2024, March). Deep learning model for arabic question-answering chatbot. In AIP Conference Proceedings (Vol. 3072, No. 1). AIP Publishing.[22]

[26] Boussakssou, M., Ezzikouri, H., & Erritali, M. (2022). Chatbot in Arabic language using seq to seq model. Multimedia Tools and Applications, 81(2), 2859-2871.[23]

[27] Abedin, A. F., Mamun, A. I. A., Nowrin, R. J., Chakrabarty, A., Mostakim, M., & Naskar, S. K. (2021). A deep learning approach to integrate human-level understanding in a chatbot. arXiv preprint arXiv:2201.02735.[24]

[28] Kumari, V., Gosavi, C., Sharma, Y., & Goel, L. (2022). Domain-Specific Chatbot Development Using the Deep Learning-Based RASA Framework. Communication and Intelligent Systems: Proceedings of ICCIS 2021, 883-896.[25]

[29] Alruily, M. (2022). ArRASA: Channel Optimization for Deep Learning-Based Arabic NLU Chatbot Framework. Electronics, 11(22), 3745.[26]

[30] Shaalan, K., Alkhatib, M., & Maghaydah, S. (2023). Machine Learning Chatbot for Sentiment Analysis of Covid-19 Tweets. CCNET, AIMLA, CICS, IOTBS, NLTM, COIT, 41-55.

[31] Ciesla, R. (2024). The Book of Chatbots: From ELIZA to ChatGPT. Springer Nature.

[32] Vajjala, S., Majumder, B., Gupta, A., & Surana, H. (2020). Practical natural language processing: A comprehensive guide to building real-world NLP systems. O'Reilly Media, Inc.

[33] Géron, A. (2017). Hands-on machine learning with Scikit-Learn and TensorFlow: Concepts, tools, and techniques to build intelligent systems (2nd ed., illustrated). O'Reilly Media.

[34] Prince, S. J. D. (2023). Understanding deep learning (Illustrated ed.). MIT Press