AN ARTIFICIAL INTELLIGENCE-BASED CHAT-BOT SYSTEM USING NATURAL LANGUAGE PROCESSING: A STUDY

Abstract

Chatbots have become a prominent means of interacting with software systems. As the demand for maintainable and scalable systems grows, there needs to be more understanding of the necessary technologies. This is evident in the scattered literature on the topic, the popularity of oversimplified building tools, and the prevalence of below conversational agents. There is a pressing need to bridge the gap between theory and practice, enabling software developers to build robust and maintainable informal systems. This paper studies the underlying techniques and technologies involved in chatbot development. To achieve this, a case study, complete with source code, is presented to explore and understand the intricacies underlying user input comprehension. A literature review that sets the context for a detailed examination of the technologies within a real-life example precedes the study. Contrary to popular belief, developing this artificial intelligence is simple enough. A modern chatbot relies on various components to achieve scalability and performance. Fortunately, many of these technologies are relatively straightforward to understand and debug for technical professionals. The chatbot processes the input text through Natural Language Processing (NLP) techniques and assigns it to a predefined intent using a classifier. We then employed another classifier to determine the appropriate actions, which may involve providing a response or executing custom software. Understanding this pipeline can prevent technical overhead when troubleshooting issues arising from a seemingly opaque black box. The insights gained from this study will be valuable for

Authors

K Sravana Kumari
Assistant Professor
Department of Computer Applications
Kakatiya Government College
Hanamkonda, Telangana, India.

Dr. B Manjula
Assistant Professor
Department of Computer Science
Kakatiya University
Hanamkonda, Telangana, India.

advancing chatbot technology, allowing developers to create more sophisticated and reliable conversational agents. By demystifying the processes involved, this paper seeks to empower software developers to confidently build chatbots, contributing to improving the chatbot landscape as a whole.

Keywords: NLP Techniques, robust, opaque black box, chatbots

I. INTRODUCTION

Artificial intelligence often seems complex and intricate to the general perception, while creating computer software that imitates human interaction can be more straightforward. This simplicity also extends to the underlying technologies used in developing intelligent systems, ranging from computationally intensive algorithms to simple and effective programs. This paper focuses on conversational agents, commonly known as chatbots, which are computer systems designed to communicate with users in a human-like manner. Chatbots engage in dialogues with users by imitating certain aspects of human communication, exchanging questions and responses to form meaningful interactions. At its core, a conversational agent should be capable of understanding a user's query and providing a sensible answer. The various ways in which this can be achieved pose exciting questions in the fields of information systems science and linguistics. The main aim of this paper is to explore the question, "How does a chatbot process language?" The core interest lies in understanding the technical actions required to bridge the gap between naturally produced language and the data stored in computer memory to enable effective dialogue between a user and a software system [1]. The challenge is to make user input understandable and computable for a deterministic machine. While an oversimplified answer to this question involves natural language processing and machine learning, the reality is more complex. Natural language processing transforms an input into a readable format, while machine learning learns the system's appropriate responses to selected queries. However, the exact steps required to understand the inner workings of conversational systems remain open for exploration.

Figure 1: AI Implemented NLP in Chatbot

Chatbots are becoming ubiquitous, and each chatbot has an engineer responsible for its development [2]. Frequently, frameworks and tools are readily available to build chatbots, leading to a surge in simple conversational agents integrated into various systems. These tools make chatbot development accessible even to those without extensive experience in text processing or software development techniques. However, when issues of scalability, optimization, and fixes arise, a deeper understanding of the technologies becomes essential. This raises questions about the knowledge of regular software engineers and data scientists who design language processing and machine learning algorithms. A significant knowledge

gap exists between technical and theoretical system-level research, which is evident in the literature. The importance of this topic is further amplified because they often hide rather than complicate the techniques and technologies involved in chatbot development. This can lead to significant overhead if they address simple issues with unnecessarily complex additions. Therefore, a thorough understanding of the underlying technologies is crucial for building better, more maintainable, optimized conversational agents.

To explore this topic, the paper undertakes a case study and an extensive literature review on the underlying technologies. The case study delves into a real-life example of a domain-specific chatbot, providing access to source code and relevant documents for a detailed examination of the technologies and techniques used. The paper is organized into various sections, starting with explaining chatbot ideas and architecture to provide a baseline understanding. It then narrows the focus to text processing techniques within chatbots by delving into natural language processing concepts in depth. This understanding is key to comprehending the case study. The paper explores the usage of these technologies within chatbots, focusing on how these techniques can be implemented into chatbot systems. This involves exploring intents, entities, dialogue structures, and different approaches to chatbots. A brief exploration of various development sites complements the literature review for chatbots, illustrating how they are built based on the concepts presented in the paper. The case study centers around Laubot, a personal assistant chatbot designed for a financial management team in a municipality's government. An external consulting organization, Gofore, carries out the development of Laubot. The case study provides a detailed look at the source code, aiming to understand how Laubot processes user input through natural language processing and decision-making algorithms.

An analysis and discussion section compares the case study with the literature review, drawing parallels between the specific example and general architectures and ideas. Additionally, a short discussion explores some problems and limitations of the study. This paper investigates into the world of chatbots, aiming to bridge the gap between theory and practice by exploring the underlying technologies used in their development. By providing insights into chatbot technology's hidden yet crucial aspects, the paper offers a pathway for building better, more efficient, and maintainable conversational agents.

II. RELATED WORKS

Abdul-Kader, S. A. et al. (2015)[3] present ChatGPT as a powerful language model, demonstrating its proficiency in generating contextually relevant and coherent responses during conversations. The model's architecture, pre-training, and fine-tuning processes contribute to its ability to understand the context and produce natural-sounding outputs. The advancements made in models like ChatGPT have significantly propelled the field of conversational AI, enhancing various human-computer interaction applications. However, responsible AI deployment is crucial to ensure such potent language models' safe and ethical use.

Braun, D. et al. (2017)[4] focus exclusively on chatbots and introduce the Transformer architecture, revolutionizing NLP and laying the groundwork for modern chatbot models. The Transformer's ability to capture long-range dependencies and contextually understand input sentences, along with its encoder-decoder architecture, has

advanced conversational AI systems. Subsequent innovations like BERT build upon the Transformer's foundations, leading to more sophisticated and context-aware chatbots offering natural and engaging user interactions.

Cahn, J. (2017) [5] offers an all-encompassing exploration of Conversational AI and Chatbots, delving into crucial aspects that underpin these technologies. The book covers dialogue systems, management principles, and various system types. It meticulously explains Natural Language Understanding (NLU), including intent recognition and entity extraction, enabling chatbots to understand user intentions effectively. The book also delves into Natural Language Generation (NLG) techniques for coherent responses. Real-world applications showcase chatbots' versatility in customer service, healthcare, and education industries. The book addresses challenges in building robust conversational agents, handling ambiguity, and ethical considerations. It discusses technological advancements, machine learning integration, and future research directions, while case studies and best practices enhance readers' understanding and expertise in crafting powerful, context-aware chatbots.

Canonico M. et al. (2018) [6] comprehensively explore deep learning techniques and architectures applied in chatbot development. The book covers essential topics, including sequence-to-sequence models, enabling chatbots to generate responses based on input sequences. Attention mechanisms allow chatbots to focus on relevant information and reinforcement learning is explored as a training method, optimizing chatbot responses based on user feedback. With a focus on cutting-edge deep learning methodologies, this book equips readers with valuable insights to build advanced and effective chatbot systems.

III. PROPOSED WORK

The Stanford CoreNLP pipeline is a powerful natural language processing toolkit that offers a wide range of tools to process and analyze natural language text [6]. The system architecture of the pipeline consists of interconnected components that work harmoniously to provide comprehensive linguistic analysis. Here is a detailed exploration of each element:

1. **Tokenization:** Tokenization is essential to natural language processing and linguistic analysis. Tokens—words, sentences, or punctuation marks—are extracted from the input text. Each token can be processed and analyzed by successive pipeline components. Tokenization breaks the text into meaningful units that a computer can understand, ready for linguistic analysis. Without tokenization, the reader would be considered a continuous string of characters, making linguistic analysis difficult.

   Tokenization usually has Numerous Steps:

   - **Word Tokenization:** The most frequent tokenization method splits input text into words. Tokenization algorithms must handle contractions, hyphenated words, and abbreviations. Word tokenization allows semantic analysis of individual words.

   - **Punctuation Tokenization:** Tokenization includes punctuation marks. Punctuation reveals text structure and meaning. The pipeline may parse punctuation marks as tokens.

- **Phrase Tokenization:** Depending on the purpose and language analysis, tokenization may identify phrases or multi-word expressions as tokens. Recognizing "New York City" as a single token is crucial for proper entity classification in named entity identification.

- **Special Tokens:** Tokenization handles URLs, emails, and numerical expressions. To avoid fragmentation, unique tokens are considered atomic units.

After tokenization, each token goes through the pipeline for linguistic analysis. Part-of-speech tagging, named entity recognition, sentiment analysis, and dependency parsing use tokenized input. Tokenization helps the Stanford CoreNLP pipeline analyze natural language text's structure, syntax, and semantics. The pipeline can extract significant insights by dividing the text into distinct tokens, enabling a wide range of natural language understanding, creation, and dialogue system applications.

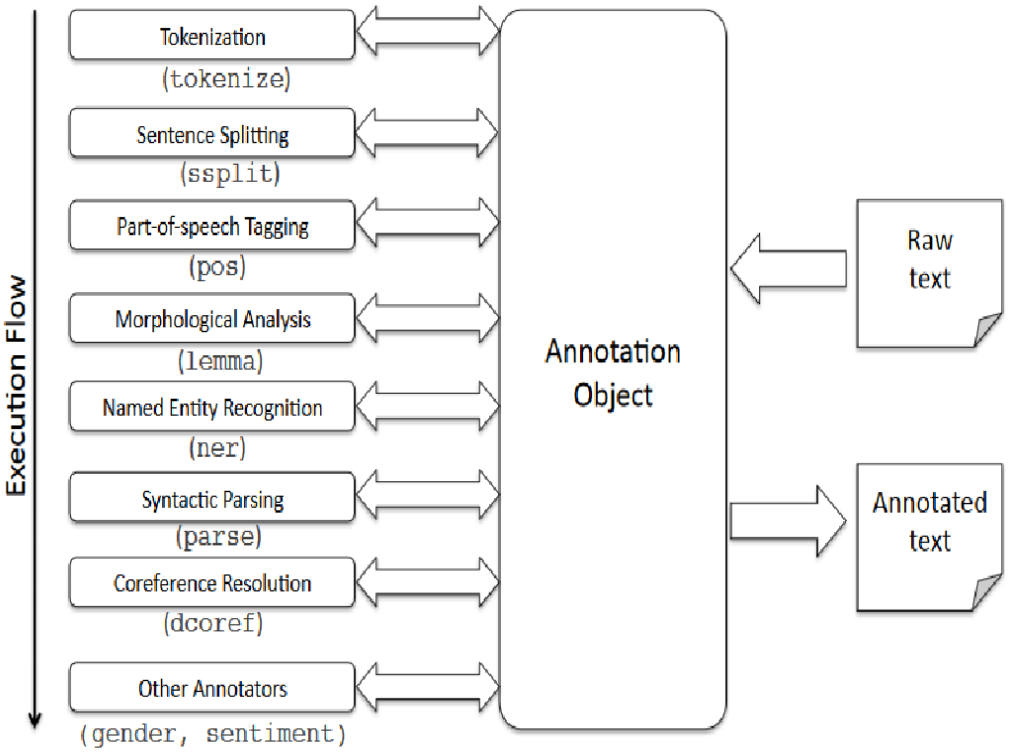

Figure 2: Stanford CoreNLP Pipeline Architecture

2. **Part-of-Speech (POS) Tagging:** Part-of-speech (POS) tagging is the next critical step in the Stanford CoreNLP pipeline after tokenization. Each token is categorized as a noun, verb, adjective, adverb, pronoun, etc. Natural language processing relies on POS tagging to clarify word meanings. POS tagging helps discern several meanings of English words.

"Bank" refers to a financial organization or tilting or inclining. POS tagging would distinguish "bank" as a noun and "went" as a verb in a sentence like "I went to the bank," providing crucial syntactic and grammatical information. Stanford CoreNLP pipeline POS tags improve named entity identification, dependency parsing, and sentiment

analysis. It helps resolve ambiguities and grasp the grammatical structure of the text.POS tagging is a crucial step in the Stanford CoreNLP pipeline that improves language understanding by providing grammatical information about each word in the input text.

3. **Named Entity Recognition (NER):** The Named Entity Recognition (NER) component in the Stanford CoreNLP pipeline plays a vital role in extracting and categorizing named entities from the input text. Named entities are specific information that refers to real-world objects, such as names of persons, organizations, locations, dates, and more. NER's primary objective is to identify these entities and assign them to predefined categories.

   For instance, in the sentence "Apple Inc. was founded by Steve Jobs on April 1, 1976, in Cupertino," the NER component would recognize "Apple Inc." as an organization, "Steve Jobs" as a person, "April 1, 1976" as a date, and "Cupertino" as a location.

4. **Dependency Parsing:** The dependency parsing component analyzes the sentence's grammatical structure and creates a dependency tree, representing the relationships between words in the sentence. This analysis aids in understanding the syntactic structure and semantic dependencies within the text.

5. **Coreference Resolution:** When pronouns or other referring expressions are employed, the Stanford CoreNLP pipeline's coreference resolution component is crucial to understanding a text. Coreference resolution connects noun phrases that refer to the same real-world thing, whether entire noun phrases or pronouns. John went to the store. Purchased groceries." "He" in the second sentence relates to "John" in the first. Coreference resolution describes the pronoun "He" to "John," guaranteeing that the proper entity is understood.

6. **Sentiment Analysis:** Sentiment analysis, also known as opinion mining, helps interpret textual data's emotions and views. Sentiment analysis classifies text as good, harmful, or neutral. This study illuminates the text's mood.

   - **Sentiment Analysis has Numerous Steps:**

     - **Text Preprocessing:** Tokenization, stop word removal, and stemming prepare the text for sentiment analysis.
     - **Sentiment Classification:** The heart of sentiment analysis is identifying text sentiment as positive, negative, or neutral. Sentiment classification often uses machine learning, deep learning, or rule-based methods.
     - **Feature Extraction:** The program must extract text features to classify sentiment. Word frequency, sentiment lexicons, and semantic word embeddings are examples.
     - Labelled datasets are utilized to train sentiment classifiers. This dataset includes sentiment-labeled text samples.

   - **Sentiment analysis is widely used in:** Sentiment analysis helps analyze social media opinions and reactions. It measures public opinion of products, services, and marketing initiatives, and Businesses analyze customer reviews using sentiment analysis. This aids customer satisfaction analysis and development, and Sentiment

analysis can track consumer sentiments towards specific companies or items, delivering marketing information [7]. Sentiment analysis helps businesses and scholars comprehend textual data's emotions and sentiments. Automating sentiment analysis helps firms make data-driven decisions, improve customer experiences, and compete in their sectors.

7. **Constituency Parsing:** A constituency parsing component is a fundamental tool in natural language processing, responsible for understanding the grammatical structure of a sentence. It generates a constituency parse tree by breaking down the sentence into its constituent parts and organizing them hierarchically, visually depicting how different parts combine to form meaningful phrases. The process involves tokenization, part-of-speech tagging, and recursive parsing using predefined grammar rules. The resulting parse tree is an intermediate representation, enabling syntactic analysis, machine translation, information extraction, and more in NLP. Its visualization aids linguists and researchers in analyzing the sentence's grammatical elements, making it a key component of advanced linguistic analysis of natural language data.

8. **Natural Logic Inference:** The natural logic inference component improves natural language processing by helping the system grasp logical relationships and inferences between sentences. It finds textual entailment and inconsistencies crucial for thinking and comprehension.

   Textual entailment is when two sentences' meanings are logically related. The natural logic inference component checks if one statement may be inferred from another. The component can deduce that "The cat is sitting on the mat" is implied by "There is a mat" if the first phrase says, "The cat is sitting on the mat," and the second sentence says, "There is a mat."

   However, natural logic inference finds sentence contradictions. One sentence contradicts another. The component prevents contradictory reactions by recognizing contradictions. The natural logic inference component detects sentence equivalence, subsumption, implication, textual entailment, and inconsistencies. Capturing these relationships improves the system's textual context-based reasoning and inferences. Natural logic inference considers the speech or document's context to draw appropriate inferences. It feels the immediate sentences and the prior and subsequent statements to understand more. Conversational AI and chatbots gain significantly from natural logic inference. It makes the system's responses coherent and consistent. The system can answer user requests by understanding textual entailment. Detecting contradictions also stops the system from presenting inaccurate information. Natural logic inference helps conversational AI systems reason and draw logical conclusions from textual input. Enhancing the system's knowledge of textual entailment and contradictions improves interactions and helps construct context-aware chatbots.

9. **Open Information Extraction:** The Open Information Extraction (OpenIE) component extracts relations and facts from text in a format suitable for database querying and knowledge base construction.

The Stanford CoreNLP pipeline's modular and flexible design allows users to customize the processing steps according to their needs. Its robust architecture and comprehensive linguistic tools make it popular for various natural language processing tasks in academia and industry. By leveraging the capabilities of these components, the pipeline provides valuable linguistic insights and enables advanced applications in natural language understanding and generation.

IV. RESULTS AND DICUSSION

The purpose of the study is to find an answer to a complex topic and to do so, a thorough literature review on chatbots and text processing was carried out. Even though the primary emphasis was placed on conversational agents, it was determined that an awareness of the many different NLP methodologies and technologies was necessary to comprehend the inner workings of contemporary chatbots. Notably, many of today's development tools for chatbots primarily rely on techniques initially developed for document processing. The human interaction research should have been purposefully included in the literature review in favor of a more comprehensive comprehension of conversational agents' technical structures and architectures [8]. This was done to ensure that the study would provide a technical perspective. Despite the efforts, the review produced many distinct approaches for achieving conversational AI, presenting few noticeable differences for a general response on how chatbots analyze language. Despite these efforts, the evaluation revealed many diverse methods to accomplish conversational AI. Despite this, particular distinctions have emerged depending on the use cases. For example, pattern-matching algorithms that use vast amounts of data are beneficial for creating general AI-imitating chatbots [9]. On the other hand, more scalable architectures that employ various text processing and machine learning algorithms are more suitable for domain-specific applications, particularly in commercial contexts.

Insights on bot technology were gleaned from a case study that focused on applying a domain-specific chatbot inside a business environment. The studied chatbot is called Laubot, and it interprets a text by using conversation pathways (stories), intentions, entities, and replies that have been manually created. Even if training data is hand-crafted, which leaves room for automation, this approach served as a basis for establishing scalability and constructing knowledge, allowing the system to grow. Custom-made training examples were found to help answer detailed and particular inquiries in an accurate and time-efficient manner. In addition, the decoupling of the fundamental data required for bot training from more advanced language processing tasks made it possible to achieve deterministic results while still being able to control them. This strategy provided a way to solve the problems associated with more generative end-to-end models and data-reliant pattern-matching chatbots [10].To transform input material written in natural language into a structured and numerical format, Laubot uses a collection of language processing tasks. These activities include tokenization, named entity recognition, and feature extraction. After the data has been analyzed, it is sent into a stack of generative feed-forward neural networks, which enables Laubot to match inputs to previously defined intents and entities. In addition, dialogue management can be accomplished by coupling a neural network with deterministic triggers, which helps mitigate the nondeterministic behavior characteristic of neural networks.

The case study and the literature review agree with one another, highlighting that contemporary development tools, particularly those developed for commercial settings, can

reap the benefits of more scalable designs, as shown by Rasa and Laubot. Because of the high degree of control exerted on the pipelines and the fact that the components may be customized, this strategy results in improved accuracy and maintainability, particularly for domain-specific knowledge-based conversational agents.

While the case study provided some helpful insights, additional research is still required to resolve some of the raised problems. The realm of conversational agents is enormous and filled with various options, making optimization difficult. Developing optimization strategies for chatbots is possible due to in-depth research on the framework's individual components. In addition, being aware of the potential uses and restrictions of chatbots can assist in determining whether they are the most appropriate answer for a particular issue. In addition, further in-depth investigations on the consequences of expanding the focus on target languages could lead to improved outcomes, mainly when advanced NLP components designed for individual languages are considered. In general, ongoing research into integrating the underlying technologies into actual chatbot creation is essential to improve chatbots' capabilities and the technology's usability.

V. CONCLUSION

This paper concludes with a technical discussion of chatbots, conversational agents. Chatbot and AI literature is extensive, yet it often needs a systematic relationship between theory and practice. This thesis connects theory and practice by evaluating Laubot's source code and documentation. Domain-specific chatbots like Laubot favor scalability over artificial intelligence. Laubot formats machine-learning data using a processing pipeline. Dialogue management generates responses using intents and entities from inputs and relevant data. This data determines the user's response. Laubot's relational database allows user-customized actions. Users can keep talking to the bot, repeating this cycle, or following prepared stories. The literature review underpins the case study's analysis. It contrasts commercial bots with intelligent assistants or chatter bots. Many chatbot applications and development tools use natural language processing as a foundation rather than a success factor. However, knowing text processing structures during chatbot development can optimize performance and uncover system flaws. This thesis combines theoretical notions with real-world chatbot development to improve the understanding of conversational agents. Laubot's implementation shows domain-specific chatbots' text-processing methodologies and architectures. The literature review also discusses chatbot creation methods and the differences between commercial usage bots and others. Researchers and developers can improve chatbot performance and usefulness by understanding chatbot theory and practice.

REFERENCES

[1] Brandtzaeg, P. B., & Følstad, A. (2017, November). Why people use chatbots. In International Conference on Internet Science (pp. 377-392). Springer, Cham.
[2] Bocklisch, T., Faulkner, J., Pawlowski, N., & Nichol, A. (2017). Rasa: Open source language understanding and dialogue management. arXiv preprint arXiv:1712.05181.
[3] 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)
[4] Braun, D., Mendez, A. H., Matthes, F., & Langen, M. (2017, August). Evaluating natural language understanding services for conversational question answering systems. In Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue (pp. 174-185).

[5] Cahn, J. (2017). CHATBOT: Architecture, design, & development. University of Pennsylvania School of Engineering and Applied Science Department of Computer and Information Science.
[6] Canonico, M., & De Russis, L. (2018). A comparison and critique of natural language understanding tools. Cloud Computing, 2018, 120.
[7] Mauldin, M. L. (1994, August). Chatterbots, tinymuds, and the turing test: Entering the loebner prize competition. In AAAI (Vol. 94, pp. 16-21).
[8] McKeown, K. (1982). The TEXT system for natural language generation: An overview. Nadeau, D., & Sekine, S. (2007). A survey of named entity recognition and classification. Lingvisticae Investigationes, 30(1), 3-26.
[9] Nivre, J., De Marneffe, M. C., Ginter, F., Goldberg, Y., Hajic, J., Manning, C. D.,... & Tsarfaty, R. (2016, May). Universal dependencies v1: A multilingual treebank collection. In Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16) (pp. 1659-1666).
[10] Plank, B., Søgaard, A., & Goldberg, Y. (2016). Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. arXiv preprint arXiv:1604.05529. Eason,

