RESEARCH ARTICLE OPEN ACCESS

Comparative Review of Open-Source Chatbot Languages: AIML and ChatScript

Prof. Rajeswari Chhualsingh, Prof. Priyadarshi Samal, Prof. Manjog Padhi
[1]*Assistant Professor, Department of MCA, Raajdhani Engineering College, Bhubaneswar, Odisha, India,*
[2]*Assistant Professor, Department of CSE, NIT, Bhubaneswar, Odisha, India,*
[3]*Research Scholar, Department of CSE, ITER, Bhubaneswar, Odisha, India,*

ABSTRACT
Human-Computer Interaction, which involves the dialogue between humans and computers, is becoming increasingly prominent in computer interaction methods. This type of program is known as a chatbot. This paper provides a review of the techniques used in chatbot design. The authors explore the similarities and differences in various chatbot implementation approaches and analyze the most popular open-source languages used for developing chatbots, specifically AIML and ChatScript. The aim of the paper is to offer a technical overview of these two languages and compare them based on parameters such as ease of implementation, language complexity, access to external resources, knowledge acquisition, integration with customized ontologies, and the potential to develop chatbots for mobile applications.

Date of Submission: 20-05-2025 Date of acceptance: 30-05-2025

I. INTRODUCTION

In recent years, there has been a significant surge in interest in chatbots. Notably, Microsoft's “Tay” bot and Facebook’s decision to integrate chatbot capabilities into Messenger accelerated the development of many new platforms. Some notable platforms include:

Microsoft Bot Framework: Microsoft announced bringing chatbot features to Skype and launched the Bot Builder for Node.js, allowing users to create their own bots.

ChatScript: Introduced in 2011 as a language for next-generation chatbots, ChatScript has won the Loebner Prize four times and offers an open-source C++ framework for developing and deploying chatbots.

Pandorabots: An online service that enables developers to build, host, and deploy chatbots using AIML (Artificial Intelligence Markup Language), an open standard, along with an open-source Java framework.

Facebook Messenger Bots: Facebook launched this tool to allow developers and businesses to create chatbots for its Messenger platform. Rebot.me: A simple service that allows usersto build, teach, and deploy chatbots on their websites.

Imperson: A chatbot generator created during Disney’s Accelerator program in 2015, supporting building, deploying, and managing chatbots across platforms like Facebook, Skype, and Twitter.

Although chatbot technology isn’t new—having been introduced nearly a decade ago with some foundational design principles—the recent surge in initiatives is driven by technological advancements. Computers are faster, smartphones are widespread, and internet services enable ubiquitous connectivity among people, data, and software. The accumulation of vast human knowledge has also opened new research avenues, especially in automatic information analysis.

According to authors [1], current chatbots are considered Partially Intelligent Systems because they demonstrate some, but not all, aspects of true intelligence. Today’s machine intelligence remains within the domain of Partial Intelligence. This raises the question: Which research areas and technologies are not yet being used optimally in current chatbots Research in Natural Language Processing (NLP) has gained momentum following the widespread adoption of commercial chatbots. However, challenges persist in understanding user intent due to the diversity in syntax and semantics. In chatbots, Natural Language Understanding (NLU) is critical—it involves extracting meaning from user input, transforming it into a semantic representation There are numerous projects focused on ontologies, knowledge organization, and knowledge representation, aiming to incorporate reasoning capabilitiesinto chatbot engines. Semantic reasoners can infer logical consequences from given facts or axioms, offering richer reasoning tools than inference engines alone. Despite these efforts, current chatbots still underutilize NLP, NLU, and advanced knowledge structuring techniques. Today, chatbots serve a variety of commercial purposes, including personal assistants, travel agents, customer support, technical support, and automatic call routing—often enhanced with speech processing technologies. Creating

AIML

According to author [8], the primary goal of AIML is to simplify the process of dialogue modeling. AIML's advantages include its stimulus-response approach and being an XML-based markup language. It defines a class object responsible for modeling conversational patterns. AIML is widely used due to its simplicity, ease of learning, straightforward implementation, and the availability of pre written AIML sets. It functions as a basic pattern-matching system, where a chatbot responds based on the connection between user questions and knowledge stored in AIML effective chatbots depends on selecting suitable technologies, which can be markup languages or scripting languages. The choice often depends on the developer's skills, project goals, and required functionalities. This study aims to identify technologies capable of supporting today's ubiquitous computing needs. The ideal solution should be open- source, easy to implement, connect to custom ontologies, and supported by an active community. It should also facilitate the development of mobile-compatible chatbot applications on Android and iOS. In this paper, we compare the two most widely used open source languages for chatbot development— analyzing their strengths and weaknesses. The choice between them depends on the specific application for which the chatbot is intended. We will provide a technical overview and compare these languages based on parameters such as ease of implementation, complexity, access to external resources, knowledge acquisition, integration with custom ontologies, and support for mobile app development. The paper is structured as follows: Section 2 covers the background of chatbot technology; Section 3 presents an overview of AIML 1.0 and 2.0; Section 4 discusses ChatScript; Section 5 compares AIML and ChatScript; and Section 6 concludes with final remarks files. Effective interaction between users and the system depends on how well this knowledge is constructed.

ontologies for reasoning capabilities, enabling chatbots to utilize and update domain knowledge via a scripting language called

AIML has both benefits and drawbacks. Its benefits include being easy to learn and implement, with a user-friendly dialogue system, and utilizing XML to represent knowledge in a formal, machine-readable way. However, it also has limitations: knowledge is stored in separate AIML files and must be manually updated if derived from internet data. Original AIML lacks extensibility, has relatively poor pattern matching capabilities, and is difficult to maintain. Although data entry is straightforward, creating a fully functional chatbot requires manually entering large amounts of content.

This paper focuses on AIML 2.0, which is a completely new version that addresses many limitations of earlier versions and aligns with the requirements of next-generation chatbots.

AIML 2.0

AIML 2.0 was motivated by the technological and social changes brought about by the mobile era. The ALICE Foundation released its specifications, aiming to improve upon earlier versions.

Ease of implementation and language complexity AIML 2.0 seeks to address previous shortcomings while maintaining the original language's simplicity. It is designed to be mostly backward-compatible with AIML 1.0, preserving ease of use. A key enhancement is improved pattern matching; besides existing wildcards _ and *, AIML 2.0 introduces #, which matches from zero to many words, allowing more concise pattern rules. Sets and maps are new features that enable knowledge integration from external files. Sets are collections of objects stored as plain text, which can be included in patterns and templates, making knowledge management more efficient. Maps are key-value pairs for representing relations, such as countries and their cities.To handle more complex interactions, AIML 2.0 introduces explicit loops via the <loop> tag, combining a Do-While and Goto functionality, allowing for better control flow. Access to external resources AIML 2.0 introduces <obb> tags, which facilitate responses from virtual assistant devices—such as making calls, sending messages, or searching the web. It also supports access to diverse online sources including Wolfram Alpha, TrueKnowledge, Answers.com, weather services, shopping sites, and other chatbots or web databases like Netbase. Netbase is a semantic web database featuring over 600 million nodes from sources like Freebase, Wikidata, and DBpedia, with built-in ontologies like WordNet. AIML 2.1 also allows creating OwlLang. A sample code illustrates how AIML can access external information—such as weather data—by querying external web services, integrating responses into chatbot interactions. Knowledge acquisition. The <learn> element allows users to teach the bot new information dynamically during conversations, storing new categories temporarily in memory. These learned categories can later be converted into AIML files using the <learnf> element, which facilitates incremental learning and updating of the chatbot's knowledge base. Tools like the Pattern Suggester in Program AB assistin semi automated pattern creation. Linking to customized ontologies Program AB—a Java based implementation of AIML—eases extension with custom tags to connect to external knowledge sources and web services. It also supports mobile and embedded systems optimizations. Developing custom ontologies for the chatbot is straightforward through tools like JENA or scripting methods—examples are shown for querying external ontologies, such as retrieving attributes of entities from background knowledge files. Building chatbots for mobile apps. The CallMom app for Android exemplifies a mobile assistant integrating multiple chatbot personalities and advanced learning capabilities. In 2013, CallMom became the first virtual assistant embedded directly on a mobile device. AIML 2.0 includes features that facilitate developing mobile chatbot applications, supporting the creation of more intelligent, responsive, and versatile mobile personal assistants.

What is CHATSCRIPT ChatScript is a scripting language designed to process user input and generate text responses. It operates in interactive volleys, similar to a game of tennis: the program takes one or more sentences from the user and responds with one or more replies. Unlike simple chatbots, ChatScript functions as a system for managing natural language interactions. It begins by transforming input words using substitution files that include mappings for common spelling mistakes, contractions, abbreviations, noise, interjections, and speech acts. These live data files are loaded at startup and are not stored permanently in the dictionary. The system also removes trailing punctuation, setting flags to identify whether the input is a question, statement, or exclamation. Since ChatScript is both a scripting language and an engine, it can be somewhat more complex to implement than AIML. A script in ChatScript is essentially a collection of rules, each comprising a type, a label, a pattern, and an output. Rules may be restricted to certain input types, such as statements, questions, or conversational gambits (indications that the chatbot is taking control of the conversation). These rules are grouped into collections called topics, which can invoke other topics to manage conversation flow. Processing an input involves matching it against pattern conditions, which may consider factors like previous dialogue history, time of day, or other criteria. When a rule's pattern is matched and its conditions are met, its output is generated—ranging from simple text to conditionalstatements, loops, or function calls. Usually, rules are executed in a specific order until one produces an output that reaches the user, at which point the response process ends. ChatScript includes a built-in WordNet lexical database, which assists with ontology and spellchecking. WordNet groups words into synonym sets (synsets), each representing a single concept, and offers brief definitions along with semantic relations between synsets. ChatScript can read structured data in JSON format from websites and can support big data or high-volume user chatbots through integration with PostgreSQL. It also offers routines for retrieving and processing data from URLs or web services, as demonstrated in its scripting examples. ChatScript can memorize specific pieces of information during a chat and store them for future use. These stored details—variables and facts—remain in long-term memory unless explicitly erased. Once set, variables can be used throughout the conversation and can trigger rules based on their values, allowing the chatbot to remember details about users and previous interactions. ChatScript has the capability to create factsin the form ofsubject verb-object triples and organize these into tables. These facts and tables can be queried to recall information. Our findings indicate that ChatScript can connect with external ontologies, retrieve knowledge from them, and incorporate this knowledge into local concept files, enabling more intelligent and context aware conversations. To develop chatbots for mobile applications, ChatScript must be embedded within another controlling program, since it is not designed as a standalone application. In this setup, the main program manages the mobile app and invokes ChatScript for conversation handling or control guidance. Because mobile devices have limited resources, the ChatScript dictionary—potentially as large as 25MB—should be minimized and optimized for mobile use to ensure efficient operation.

Comparative Analysis

The primary objective of this study was to identify the most suitable technology for developing a personal chatbot. To achieve this, we first selected chatbot languages based on open-source platforms. The next step involved comparing these languages according to specific requirements outlined in the introduction.

Table 1 compares AIML and ChatScript

based on several parameters: ease of implementation, simplicity, ability to connect with customized ontologies, support from the open-source community, and suitability for developing mobile applications on Android and iOS platforms.

ChatScript was initially designed to provide natural language understanding capabilities for game characters but has also been effectively utilized for chatbot development. It is a meaning-based system that supports sets of words called concepts, which are used to representsynonyms. If the goal isto create the illusion that the chatbot understands the user—minimizing instances where it responds inappropriately and maximizing accurate, contextually relevant replies—then ChatScript is a suitable choice. AIML (Artificial Intelligence Markup Language) is a widely adopted markup language for designing chatbots. AIML 2.0 introduces numerous new features, including: The <topic> tag can now be assigned at the category level to facilitate organizing categories into topics. New wildcards (^ and #) that match zero or more words, enhancing pattern flexibility. The - The <topic>` tag can now be assigned at the category level to facilitate organizing categories into topics. New wildcards (^ and #) that match zero or more words, enhancing pattern flexibility. The pattern marker to assign highest priority to certain patterns. Attribute tags allowing any template tag attribute to be set using sub elements. The <set> tag for evaluating patterns based on predefined word sets. The <map> tag for lookup operations within predefined mappings, returning corresponding values. Wildcards in condition patterns for default matching. The <loop> tag to enable repeated execution of conditional statements. The <var> attribute to define local variables scoped to categories. The <sraix> tag to handle remote requests to other chatbot instances or services. <normalize> and <denormalize> tags for character conversion. Formatting options for date handling. <request> and <response> tags to manage prior user inputs and bot responses. <learn>, <learnf>, and <eval> tags for dynamic learning. <explode> for splitting words into individual characters. The <oob> (out-of-band) tag to support commands for clients and mobile devices. Program AB, an implementation of AIML 2.0, incorporates features such as Sets, Maps, wildcards, and the ability to connect to remote bots and web services via new tags. It also includes memory optimization techniques, making it possible to run a chatbot on mobile devices and embedded systems. Program AB has been successfully deployed on Android smartphones and tablets. If the goal is to develop a customized chatbot that meets the technical requirements for ubiquitous connectivity among people, information, and software— while adapting to the social changes driven by the mobile era—then AIML 2.0 presents a logical choice.

II. CONCLUSION

In this paper, we provided a technical overview of the two most widely used open-source languages for building chatbots. We compared these languages based on several parameters, including ease of implementation and language complexity, access to external resources, methods of knowledge acquisition, integration with customized ontologies, and the feasibility of developing chatbots for mobile applications. The AIML 2.0 specification introduces new features while maintaining simplicity, making it particularly accessible for non-programmers.

From the survey presented above, it can be concluded that technologies for chatbot development and implementation remain diverse, with no universally accepted approach currently established. The choice of technology largely depends on the developer's preferences and decisions.

Analysis of existing chatbots indicates a need for design improvements, such as advanced Natural Language Processing (NLP) and Natural Language Understanding (NLU), enhanced pattern recognition techniques, more comprehensive knowledge bases, and better knowledge organization and representation. Additionally, further research should explore the use of personalized, customized ontologies thatincorporate reasoning capabilities and can establish specific personality traits by recording behavioural patterns from conversation histories.

REFERENCE

Journal Articles

[1]. A. Khanna, B. Pandey, K. Vashishta, K. Kalia, B.Pradeepkumar and T. Das, "A Study of Today's A.I. through Chatbots and Rediscovery of Machine Intelligence", IJUNESST, vol. 8, no. 7, pp. 277-284,2015.

[2]. A. TURING, "I.—COMPUTING MACHINERY ANDINTELLIGENCE", Mind, vol., no. 236, pp. 433-460,1950.

[3]. Shawar, Bayan Abu, and Eric Atwell. "Chatbots: are they really useful?." In LDV Forum, vol. 22, no. 1, pp. 29-49.2007

[4]. Hutchens, Jason L.; Alder, Michael D. (1998),"Introducing MegaHAL" (PDF), NeMLaP3 / CoNLL98Workshop on

Human Computer Conversation, ACL(271): 274

[5]. Batacharia, B., D. Levy, R. Catizone, A. Krotov, and Y.Wilks. "CONVERSE: a conversational companion." In Machine conversations, pp. 205-215. Springer US, 1999

[6]. Abu Shawar B. and Atwell E. 2002. A comparison between ALICE and Elizabeth chatbot systems. School of Computing research report 2002.19, University of Leeds

[7]. HEXBOT chatbot website. http://www.hexbot.com/

[8]. Richard S. Wallace, The Elements of AIML Style,ALICE A. I. Foundation, Inc. March 28, 2003

[9]. M. Bruno Marietto, R. Aguiar, G. Barbosa, W.Botelho, E. Pimentel, R. Franca and V. da Silva, "Artificial Intelligence Markup Language: A Brief Tutorial", International Journal of Computer Science &Engineering Survey, vol. 4, no. 3, pp. 1-20, 2013.

Books & Book Chapters

[10]. Wallace, R. S. (2014a). AIML 2.0 Working Draft,https://docs.google.com/document/d/1 wNT25hJRyupcG51aO89UcQEiG -HkXRXusukADpFnDs4/pub

[11]. (Wallace, AIML - Sets and Maps in AIML 2.0,https://docs.google.com/document/d/1D WHiOOcda58CflDZ0Wsm1CgP3Es6dpic b4MBb bpwzEk/pub

[12]. NetBaseAPIdocumentation,(https://market.mashape.com/pannous/netbase)

[13]. Lundqvist KO, Pursey G, Williams S, “Design and implementation of conversational agents for harvesting feedback in eLearning systems”. In: Hernandez-Leo D,Ley T, Klamma R, Harrer A (eds) Scaling up learning for sustained impact. Lecture notes in computer science, vol8095, pp 617–618., 2013

[14]. AIML 2.0 Reference, http://callmom.pandorabots.com/static/refer ence/#elements/-lt-learn-gt)

[15]. M. McTear, C. Zoraida, and G. David. "Conversational Interfaces: Devices, Wearables, Virtual Agents, and Robots." In The Conversational Interface, pp. 283-308Springer International Publishing, 2016Conference Proceedings

[16]. Bruce Wilcox, ChatScript Basic User’s Manual,Revision11/18/2013 cs3.73,http://chatscript.sourceforge.net/Docu mentation/ChatScript%20Basic%20User%20M anual.pdf

[17]. Ramon López de Mántaras Badia, ARBOR Ciencia, Pensamiento y Cultura, Vol. 189-764, no 2013, a086ISSN-L: 0210-1963, doi:http://dx.doi.org/10.3989/arbor.2013.764 n60 09)Technical Reports Online Sources

[18]. What's new in AIML 2.0,https://www.botlibre.com/forum post?id=1156738