International Journal of Computational Research in Science and Technology
E-ISSN: 3108-0871, Volume 2 Issue 5
Sep-Oct- 2026, PP 01-7

AI-Based Smart Assistants: A Review Of Natural Language Understanding, Task Automation And Multimodal Interaction

Dhokare Apeksha Chimaji[1] Prof. N. G. Rokade[2] Prof. M. C. Kolse[3] Prof. S. K. Said[4]
Department of Artificial Intelligence and Data Science
Jaihind College of Engineering
Pune, Maharashtra, India

Abstract—Smart assistants help users complete everyday tasks such as answering questions, searching information, setting reminders, checking the weather, opening applications and controlling devices through natural voice or text commands. This paper reviews the technologies that make such assistants possible and the way they have evolved from rule-based chatbots to systems built on large language models that can call external tools. The review covers input processing through speech recognition and text normalization, natural language understanding with intent classification and slot filling, dialogue management, task execution through application programming interfaces and operating-system commands, and response generation through text and speech synthesis. Retrieval-augmented generation and tool-use methods that ground assistant answers in current information are examined, along with personalization and memory. A modular architecture suitable for implementation in Python is proposed, covering the interaction, understanding, reasoning, skills and integration layers. A research gap analysis, objectives, a comparison of assistant design paradigms and a discussion of challenges in accuracy, connectivity, privacy and multilingual support are presented. The review concludes that combining a reliable task-oriented core with a language-model reasoning layer offers an efficient, user-friendly and extensible assistant for education, healthcare, business and home automation.

Index Terms—Smart assistant, artificial intelligence, natural language processing, intent classification, large language models, tool use, task automation, human–computer interaction

I. Introduction

1.1 Motivation

The idea of conversing with a computer in everyday language dates back to ELIZA [1], a pattern-matching program that imitated a psychotherapist. Six decades later, smart assistants are present on phones, computers, cars and household speakers, and they perform useful work rather than merely imitating conversation. Hoy [2] describes how assistants such as Siri, Alexa, Cortana and Google Assistant became mainstream by combining speech recognition with cloud services.

A smart assistant reduces the effort required to complete routine activities. Instead of opening several applications to check the weather, add a calendar entry and play music, a user can express each need in a sentence. The recent arrival of large language models (LLMs) [3] has expanded what assistants can understand, allowing open-ended questions and multi-step requests.

This paper reviews the design of AI-based smart assistants, with a focus on understanding user intent and executing tasks reliably. Its contributions are a survey of key techniques, a modular reference architecture, an analysis of research gaps and a discussion of challenges.

1.2 Objectives of the Review

The objectives of this review are:

International Journal of Computational Research in Science and Technology
E-ISSN: 3108-0871, Volume 2 Issue 5
Sep-Oct- 2026, PP 01-7

• To study the components of AI-based smart assistants, from input processing to response generation.

• To compare rule-based, task-oriented and LLM-based assistant paradigms.

• To propose a modular architecture implementable with Python libraries.

• To examine methods for grounding answers and executing tasks reliably.

• To identify challenges and future directions such as multilingual support and IoT integration.

II. Overview of Key Concepts

2.1 Assistant Paradigms

Assistants can be grouped into three paradigms. Rule-based assistants match keywords or patterns to fixed responses. Task-oriented assistants classify the user's intent and extract parameters (slots), then call a skill to fulfil the request. LLM-based agents use a language model to interpret the request, plan steps and call tools. Fig. 1 summarizes the core capabilities of a smart assistant.

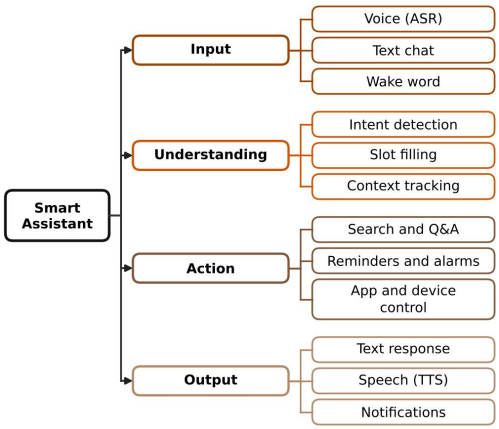


Fig. 1. Core capabilities of an AI-based smart assistant.

2.2 Natural Language Understanding

Spoken language understanding is traditionally split into intent detection and slot filling [4]. For the request “remind me to submit the report at five pm,” the intent is set_reminder and the slots are task = “submit the report” and time = “17:00”. Transformer encoders such as BERT [5] improved both tasks, and Chen *et al.* [6] showed that a single BERT model can learn them jointly.

III. Review of Existing Work

Weizenbaum's ELIZA [1] demonstrated that even shallow pattern matching can create an impression of understanding, but it also revealed the limits of rule-based systems that lack real knowledge. López *et al.* [7] compared commercial assistants on a common set of tasks and found large differences in their ability to handle follow-up questions and complex requests, which highlighted the importance of dialogue context.

Speech is the most natural input. Deep neural network acoustic models [8] dramatically reduced recognition errors, and the Whisper model [9], trained on a very large and diverse set of audio, provides robust multilingual transcription that is valuable for Indian users who mix languages. On the output side, Tacotron 2 [10] produces natural-sounding speech from text.

For understanding, Chen *et al.* [6] achieved strong joint intent and slot accuracy with BERT. Large language models trained to follow instructions with human feedback [11] can understand loosely phrased requests and respond helpfully without task-specific training data.

LLMs alone, however, can produce outdated or fabricated answers. Retrieval-augmented generation (RAG) [12] addresses this by retrieving relevant documents and conditioning the answer on them. ReAct [13] interleaves reasoning steps with actions such as web searches, and Toolformer [14] showed that a model can learn when to call external tools like calculators or calendars. Together these methods allow an assistant to act on current information and perform real tasks. Table I summarizes the reviewed literature.

Ref.	Authors	Year	Technique Used	Merits	Demerits
[1]	Weizenbaum	1966	ELIZA pattern-matching chatbot	First conversational program	No understanding or task execution
[7]	López et al.	2017	Comparison of Siri, Alexa, Cortana, Google	Differences in handling complex requests	Small task set
[2]	Hoy	2018	Overview of commercial voice assistants	Explained capabilities and adoption	Descriptive; no evaluation
[6]	Chen et al.	2019	Joint BERT intent + slot model	High accuracy on benchmark datasets	Needs labelled domain data
[12]	Lewis et al.	2020	Retrieval-augmented generation	Grounds answers in retrieved documents	Depends on retrieval quality
[11]	Ouyang et al.	2022	Instruction tuning with human feedback	Helpful responses to open requests	Can still produce errors
[9]	Radford et al.	2023	Whisper weakly supervised ASR	Robust multilingual transcription	Large model; latency on CPUs
[13]	Yao et al.	2023	ReAct reasoning and acting	Interleaves reasoning with tool calls	Long chains add latency
[14]	Schick et al.	2023	Toolformer self-supervised tool use	Model learns when to call APIs	Limited set of tools

TABLE I Summary of Literature Survey

IV. Identified Research Gaps

The following gaps are identified:

- **G1:** Many academic assistants support only a fixed command list and fail on paraphrased or multi-part requests.
- **G2:** LLM-based assistants can hallucinate or take incorrect actions without confirmation steps.
- **G3:** Support for Indian languages and code-mixed speech remains limited in open-source tools.

International Journal of Computational Research in Science and Technology
E-ISSN: 3108-0871, Volume 2 Issue 5
Sep-Oct- 2026, PP 01-7

- **G4:** Most assistants depend on continuous internet access, with few offline fall-backs.
- **G5:** User data such as voice recordings and reminders is often processed in the cloud without clear privacy controls.

V. System Architecture

The proposed architecture, shown in Fig. 2, combines a deterministic task core with an LLM reasoning layer, so that common commands are executed quickly and predictably while unusual requests benefit from the flexibility of a language model.

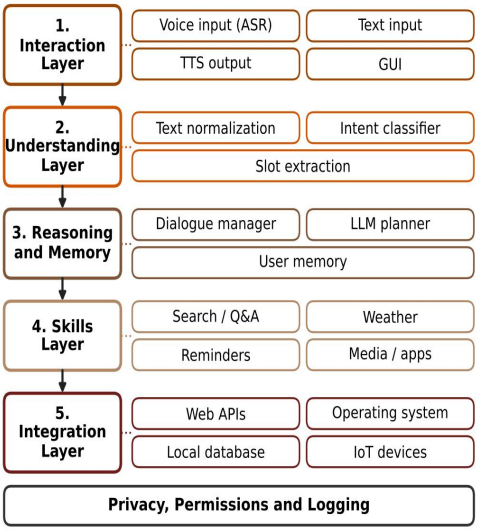


Fig. 2. Proposed hybrid architecture of the AI-based smart assistant.

5.1 Interaction Layer

Users interact by voice or text through a desktop or mobile interface. Voice input is converted to text by a speech-recognition engine, and responses are spoken back using text-to-speech.

5.2 Understanding Layer

Text is normalized (lower-casing, spelling correction, number and date parsing), and an intent classifier with slot extraction identifies what the user wants. If confidence is low, the request is forwarded to the reasoning layer.

5.3 Reasoning and Memory Layer

A dialogue manager tracks context across turns. An LLM interprets complex requests, decides which tools to call and composes answers. Short-term memory stores the current conversation, and long-term memory stores user preferences such as preferred city for weather.

5.4 Skills Layer

Skills are modules for specific tasks: web search, weather, reminders, alarms, music, application launching, calculations and smart-home control.

International Journal of Computational Research in Science and Technology
E-ISSN: 3108-0871, Volume 2 Issue 5
Sep-Oct- 2026, PP 01-7

5.5 Integration Layer

Skills connect to external APIs, the operating system, a local database and IoT devices. Sensitive actions require explicit confirmation.

VI. Implementation Methodology

The processing of a request follows the workflow in Fig. 3. The intent classifier outputs a probability distribution over intents; the predicted intent is

$$\hat{\imath} = \operatorname{argmax}_i P(i \mid x) \quad (1)$$

where x is the input text. The request is handled by the task core if $P(\hat{\imath} \mid x)$ exceeds a confidence threshold τ; otherwise the LLM layer is invoked. Performance can be measured by intent accuracy, slot F1-score, task completion rate and average response time.

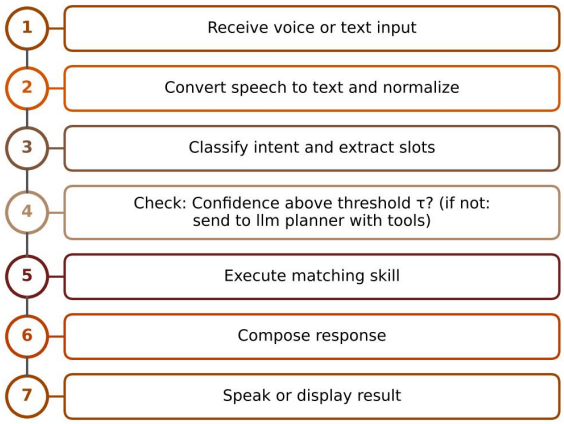


Fig. 3. Request handling workflow of the smart assistant.

A typical Python implementation uses a speech-recognition library for input, a text-to-speech engine for output, a lightweight transformer for intent classification, and HTTP clients for weather and search APIs.

VII. Performance Comparison

Table II compares the three assistant paradigms. Rule-based systems are fast and predictable but rigid; task-oriented systems are accurate within their domain; and LLM-based agents are flexible but slower and need safeguards. A hybrid approach captures the advantages of each.

Paradigm	Flexibility	Reliability	Speed
Rule-based	Low	High	Very fast
Task-oriented (intent + slots)	Medium	High	Fast
LLM agent	Very high	Medium	Slower
Hybrid (proposed)	High	High	Fast for common tasks

TABLE II Comparison of Assistant Paradigms

VIII. Open Issues and Future Directions

Table III lists the main challenges. Future assistants are expected to be multilingual and multimodal, accepting images and documents as well as speech, to run smaller language models on-device for privacy and offline use, and to integrate with IoT devices for home and campus automation. Accessibility benefits for people with disabilities [15] make inclusive design a priority.

Challenge	Mitigation
Speech errors and accents	Robust ASR; confirmation of key slots
Hallucinated answers	Retrieval grounding; cite sources
Internet dependence	On-device intent model; offline skills
Privacy of voice data	Local processing; user data controls
Multilingual users	Multilingual ASR and NLU models

TABLE III Challenges and Mitigation

IX. Conclusion

AI-based smart assistants have progressed from pattern matching to systems that understand natural requests and act on them. The reviewed literature shows that robust speech recognition, transformer-based intent understanding and tool-using language models together make assistants genuinely useful. A hybrid architecture that executes common tasks through a reliable skill core while using an LLM for complex requests, with confirmation for sensitive actions and clear privacy controls, provides an efficient, user-friendly and intelligent solution for simplifying everyday activities.

9.1 Acknowledgment

The authors express their sincere gratitude to the Department of Artificial Intelligence and Data Science, Jaihind College of Engineering, Pune, for providing the facilities, guidance and encouragement needed to carry out this work. The authors also thank the faculty members and colleagues whose valuable suggestions helped improve the quality of this review.

References

1. J. Weizenbaum, “ELIZA—A computer program for the study of natural language communication between man and machine,” Commun. ACM, vol. 9, no. 1, pp. 36–45, 1966.
2. M. B. Hoy, “Alexa, Siri, Cortana, and more: An introduction to voice assistants,” Med. Ref. Serv. Q., vol. 37, no. 1, pp. 81–88, 2018.
3. T. Brown et al., “Language models are few-shot learners,” in Proc. NeurIPS, vol. 33, 2020, pp. 1877–1901.
4. G. Tur and R. De Mori, Spoken Language Understanding: Systems for Extracting Semantic Information from Speech. Chichester, U.K.: Wiley, 2011.
5. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in Proc. NAACL-HLT, 2019, pp. 4171–4186.
6. Q. Chen, Z. Zhuo, and W. Wang, “BERT for joint intent classification and slot filling,” arXiv:1902.10909, 2019.

7. G. López, L. Quesada, and L. A. Guerrero, “Alexa vs. Siri vs. Cortana vs. Google Assistant: A comparison of speech-based natural user interfaces,” in Proc. AHFE, Springer, 2017, pp. 241–250.
8. G. Hinton et al., “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” IEEE Signal Process. Mag., vol. 29, no. 6, pp. 82–97, 2012.
9. A. Radford et al., “Robust speech recognition via large-scale weak supervision,” in Proc. ICML, 2023, pp. 28492–28518.
10. J. Shen et al., “Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions,” in Proc. IEEE ICASSP, 2018, pp. 4779–4783.
11. L. Ouyang et al., “Training language models to follow instructions with human feedback,” in Proc. NeurIPS, vol. 35, 2022, pp. 27730–27744.
12. P. Lewis et al., “Retrieval-augmented generation for knowledge-intensive NLP tasks,” in Proc. NeurIPS, vol. 33, 2020, pp. 9459–9474.
13. S. Yao et al., “ReAct: Synergizing reasoning and acting in language models,” in Proc. Int. Conf. Learn. Represent. (ICLR), 2023.
14. T. Schick et al., “Toolformer: Language models can teach themselves to use tools,” in Proc. NeurIPS, vol. 36, 2023.
15. A. Pradhan, K. Mehta, and L. Findlater, ““Accessibility came by accident”: Use of voice-controlled intelligent personal assistants by people with disabilities,” in Proc. CHI, 2018, Paper 459.