IET Intelligent Transport Systems

WILEY

IET The Institution of Engineering and Technology

ORIGINAL RESEARCH OPEN ACCESS

ReChat: A Task-Based Chatbot for EV Charging Management Optimization

Pablo Donate[1] | Julio A. Sanguesa[1] | Piedad Garrido[1] | Vicente Torres-Sanz[1] | Francisco J. Martinez[1] | Carlos T. Calafate[2]

[1]Department of Computer Science and System Engineering, University of Zaragoza, Zaragoza, Spain | [2]Department of Computer Engineering, Universitat Politècnica de València, València, Spain

Correspondence: Pablo Donate (pdonate@unizar.es)

Received: 7 November 2025 | **Revised:** 24 February 2026 | **Accepted:** 8 March 2026

ABSTRACT

The increasing adoption of electric vehicles (EVs) and the evolution of connected vehicle systems have led to a growing need for intelligent charging management solutions. This article introduces ReChat, a task-based multilingual chatbot designed to optimize EV charging management through reliable task-oriented intent understanding and safe action dispatch. By leveraging natural language processing (NLP), ReChat enables seamless user interaction with charging systems across six languages (Spanish, English, German, French, Italian and Portuguese). A custom dataset was developed to train and evaluate the chatbot's intent-classification capabilities, ensuring robust performance in diverse linguistic contexts. A comparative analysis of multilingual Bidirectional Encoder Representations from Transformers (mBERT)-based intent classifiers shows that a single pooled multilingual mBERT model achieves macro-$F1$ values between 68.0% and 78.4% across languages, while language-specific mBERT models yields 64.7% to 78.9%. This work advances the development of robust conversational AI systems for smart transportation, outlining a modular architecture and empirical evidence on multilingual adaptability in real-world applications.

1 | Introduction

The mitigation of climate change and the reduction of carbon emissions have led to the adoption of electric vehicles (EVs) worldwide [1, 2]. According to the International Energy Agency, electric vehicle sales are projected to reach 40% by 2030 under current policies [2]. This growth highlights the commitment to sustainable mobility, but presents challenges related to the efficient management of charging these vehicles, including technical complexity, heterogeneity of standards, interoperability and variability in charging times [3–5].

This growth creates practical challenges for managing charging efficiently, especially when user interaction is required. Traditional interfaces (dedicated screens or mobile apps) can make charging configuration cumbersome, requiring users to navigate multi-screen menus for routine operations such as pausing or resuming a session, checking status or accessing recent charge history. For users who wish to customize advanced parameters, vendor applications often present these settings through deeply nested interfaces that assume familiarity with electrical terminology and charger state models. In connected and automated-vehicle settings, reducing interaction friction becomes even more important. Conversational interfaces are a natural candidate because they can expose both routine charging operations and optional advanced settings through simple language [6].

Recent advancements in natural language processing (NLP) and its application to conversational reasoning have enabled the development of chatbots that support more context-aware natural-language interaction [7]. These tools have proven effective in various domains, from customer service to education and healthcare, and their application in vehicular technology can improve accessibility to complex services in the Internet of

Things (IoT) and smart transportation systems [8]. Specifically, a chatbot designed for EV charging management can simplify routine charging control for all users (e.g., starting, pausing or checking session status through natural-language commands), while providing a more accessible interface for advanced users who already configure parameters such as current setpoints or charging schedules in vendor applications [9].

Selecting the appropriate technology to implement a chatbot with NLP capacity involves multiple considerations, including precision, flexibility, scalability and the ability to adapt to multilingual environments. There are several different options available, ranging from conversational platforms with support for advanced models to highly configurable deep learning models. Evaluating these technologies in terms of multilingual adaptability is critical to ensure effective user interaction in diverse linguistic contexts. Performing comparative assessments is important to guide the selection of the most suitable technology for EV charging management in connected and automated vehicle systems.

This article presents the design and development of ReChat, a task-based multilingual intelligent chatbot aimed at optimizing EV charging management. This assistant combines NLP to offer users natural-language interaction with their charging devices, while maintaining a modular architecture for the incorporation of new NLP models. A comparative analysis of chatbot implementation technologies, including platform-based solutions and transformer-based intent classifiers (language-specific mBERT vs. pooled multilingual mBERT), identified the most suitable solution for this use case, considering factors like intent classification accuracy, multilingual capability and ease of integration.

This work contributes to the fields of vehicular technology and human-machine interaction by presenting an intelligent chatbot and offering a systematic methodology for selecting appropriate technologies in similar use cases. By addressing the challenges, this study advances the development of task-based dialogue systems for real-world applications in smart transportation. Guidelines are established for future projects that combine technical complexity and linguistic diversity, aiming to maximize operational effectiveness and user experience.

The contributions of this work are fourfold: (i) a task-to-action ontology for EV charging control that maps user intents to auditable application programming interface (API) calls; (ii) a multilingual dataset in six European languages, balanced by intent, together with a validation protocol based on large language model (LLM)-generated test sets; (iii) a comparative study of platform-based solutions versus mBERT-based intent classifiers (language-specific mBERT vs. pooled multilingual mBERT) for EV charging intents and (iv) an end-to-end Telegram prototype demonstrating the practical feasibility of our proposal, enabling essential charging operations accessible to all users (charger state and live consumption, pause/resume, enable/disable, recent-charge statistics, device listing and language management), as well as optional advanced configuration (current setpoint adjustment, schedule management) for users who already manage these parameters in vendor applications.

The subsequent sections of the article are structured as follows. In Section 2, related work on chatbots with NLP is reviewed. Section 3 describes the proposed solution, detailing key functionalities and selected technologies. Section 4 outlines the methodology, datasets created and training processes. In Section 5, we present the comparative evaluation results. Finally, Section 6 concludes with the main contributions, limitations and future research directions.

2 | Related Work

We organize the related literature into three streams: (i) task-oriented chatbots for domain services and their typical NLP back-ends (intent classification vs. platform-based solutions), (ii) recent LLM-based task dialogue and tool-use paradigms and their trade-offs with deterministic pipelines and (iii) EV charging management and optimization works that motivate the operational goals and user-facing control actions exposed by ReChat.

Task-oriented chatbots have matured into a practical interface layer for domain services, enabling users to accomplish concrete goals through natural language while the system executes constrained actions. In most deployments, the core design choice is how to translate free-form utterances into executable operations. Traditional pipelines rely on intent classification and slot filling with scripted policies, whereas platform-based solutions provide graphical flows and embedded natural language understanding modules. These approaches trade off controllability and auditability against development speed and language coverage. Deterministic execution facilitates safety and debugging, but performance depends strongly on labelled diversity and clear intent boundaries. Conversely, low-code platforms accelerate prototyping and integration, yet multilingual robustness can degrade when training data are limited or when the underlying embeddings are English-centric.

Within this stream, recent works illustrate the breadth of task-oriented assistants and the recurring architectural pattern of mapping intents to actions. For instance, task-based chatbots have been used for institutional information access and service automation in university environments [10], while agentic dialogue management with memory components has been explored for multi-topic task conversations [11]. Fine-tuned transformer encoders such as BERT have also been shown to be effective for intent recognition in domain-specific assistants [12]. In parallel, low-code conversational platforms can speed up deployment and multi-channel integration [13], and visual flow design has demonstrated practical benefits in task-oriented deployments [14].

The architectural pattern of intent-to-action mapping via supervised classifiers extends across multiple vertical domains, each with distinct operational constraints that shape their design priorities. In healthcare, multilingual symptom-checking and triage assistants prioritize safety constraints and explicit confirmation policies to prevent misdiagnosis or inappropriate treatment recommendations [15]. These systems often combine intent classification with rule-based validation to ensure that critical medical advice is never dispatched under uncertainty [16]. In e-commerce and customer support, multilingual query-understanding bots emphasize transactional correctness and explicit slot-filling for

order management, product search and refund processing [17]. In public services and government settings, multilingual virtual assistants for citizen inquiries prioritize transparency, auditability and regulatory compliance, requiring logged decision trails and version-controlled intent models to meet accountability standards [18]. These domain-specific deployments share a common emphasis on controllability and deterministic execution, while differing in their operational context: healthcare bots address liability and patient safety, e-commerce bots optimize for transaction accuracy and fraud prevention and government bots enforce procedural transparency. ReChat extends this family of systems to the EV charging domain, where the safety-critical nature of controlling physical charging equipment and grid connections motivates a similarly deterministic and auditable execution regime.

More recently, task-oriented dialogue has been influenced by LLMs that can plan and interact with external tools. Modular approaches such as MRKL-style systems (*modular reasoning, knowledge and language*) combine specialized modules (e.g., classifiers, retrievers and calculators) with a language interface to improve controllability and compositionality in tool-augmented tasks [19]. ReAct-like prompting (*reasoning and acting*) interleaves reasoning and action steps, enabling an LLM to decide when to call tools or APIs during multi-step tasks [20]. Toolformer-style training further explores how language models can learn to invoke tools from weak supervision, broadening the range of tasks that can be executed through tool calls [21].

These LLM-centric paradigms provide flexibility and rapid feature iteration, but they also motivate clearer engineering trade-offs for safety-critical domains: (i) deterministic, auditable intent-to-action mappings reduce the risk of unintended operations, (ii) on-prem supervised intent classifiers avoid third-party inference dependencies and (iii) explicit parameter validation and confirmation policies are easier to enforce when execution is decoupled from free-form generation. ReChat follows this classification-based paradigm for action dispatch, while remaining compatible with future hybrid extensions (e.g., LLM-assisted disambiguation that still routes through a fixed action layer).

Complementing the chatbot literature, EV charging management optimization has matured around scheduling, coordinated control and grid-aware constraints. Multi-objective scheduling at charging stations with on-site renewable generation highlights the need to jointly optimize operational performance and site/grid constraints under energy-availability variability [22]. User-centric charging objectives (e.g., priority rules and service differentiation at fast-charging stations) are commonly addressed via coordinated schemes that incorporate demand-based priorities and operational constraints [23]. Grid- and site-centric objectives (e.g., mitigating phase imbalance and improving power quality under stochastic EV arrivals) motivate control policies that explicitly account for load imbalance and network-side impacts [24]. Tariff- and cost-aware objectives in residential or depot scenarios are often studied through comparative analyses of real-time coordinated charging schemes, highlighting practical trade-offs across control strategies [25]. At the power-system level, the interaction between EV charging demand and decarbonization objectives has also been studied through optimal economic dispatch formulations that incorporate the social cost of emissions, reinforcing the relevance of cost- and policy-aware charging strategies [26].

Beyond optimization formulations, several systems demonstrate end-to-end orchestration by integrating smart-charging logic with charger back-ends and user-facing components. For example, fleet-oriented smart-charging extensions coordinate sessions under grid and operational constraints [27], and laboratory-validated coordinated charging demonstrations highlight realistic hardware-in-the-loop integration and control [28]. These systems underline the practical value of exposing charging control through well-defined operations over normalized charger APIs, an integration pattern that motivates our intent-to-action design on the conversational interface layer.

To consolidate the discussion above, Table 1 provides a systematic summary of the three streams along explicit design axes (dialogue paradigm, execution layer and safety/controllability). We then position ReChat on these axes in the subsequent paragraph.

On these axes, ReChat follows a task-oriented dialogue paradigm with supervised intent classification, a deterministic intent-to-API execution layer, and a safe-by-default policy (authentication and user–device binding, state/bounds checks and clarification/confirmation under uncertainty).

EV charging management typically targets three operational goals. First, user-centric objectives include meeting departure targets, session monitoring and cost transparency. Second, grid- and site-centric objectives include peak shaving, demand response and feeder or breaker constraints. Third, tariff-aware objectives focus on cost minimization under time-of-use or dynamic pricing. Common control levers include start/pause/resume, enable/disable, current setpoint (intensity), schedule creation and device selection. In practice, these actions are orchestrated through standardized protocols (e.g., open charge point protocol [OCPP]) or vendor APIs, and they can be combined with optimization modules such as smart charging or load balancing. Our work focuses on the conversational interface and safe action dispatch, which complements optimization engines.

We did not identify prior work that jointly provides: (i) an end-to-end task-based conversational interface covering common EV charging control and monitoring actions (e.g., pause/resume, enable/disable, intensity setpoint, schedules, device selection and recent-charge statistics), and (ii) multilingual intent understanding evaluated across multiple languages within a single, auditable execution pipeline. This motivates ReChat as a task-based, multilingual assistant that maps charging-related intents to deterministic back-end operations under explicit safety and controllability constraints [29].

Recent multilingual encoders and sequence-to-sequence models report strong cross-lingual transfer and are compatible with our pipeline. For example, XLM-R-based approaches have been used successfully in multilingual classification settings [30]. Stronger multilingual backbones such as mDeBERTa can further improve cross-lingual generalization under distribution shift [31]. Sentence-embedding models such as LaBSE have also been optimized for multilingual semantic tasks and remain relevant alternatives for intent representations [32]. We prioritize compact

TABLE 1 | Systematic summary of related work along explicit design axes and positioning of ReChat.

Stream	Dialogue paradigm	Execution layer and safety/controllability	Representative works
Task-oriented chatbots	Task-oriented, intent/slot-based dialogue	Deterministic intent-to-action execution; auditable flows; safety depends on data quality and intent separation	[10–18]
LLM tool-use paradigms	LLM-driven, agentic dialogue with tool use	Dynamic tool/API calls; flexible but harder to audit; requires guardrails and confirmations for safe execution	[19–21]
EV charging management	Optimization-centric control (user-, grid-, tariff-aware)	Backend schedulers and coordinators over OCPP/vendor APIs; limited user-facing control; safety via explicit constraints	[22–28]

transformer encoders for controllable supervised intent classification under tight data/compute budgets, predictable latency on modest hardware and straightforward on-prem deployment; swapping the encoder for stronger multilingual backbones remains straightforward future work within the same code path.

3 | ReChat: An Intelligent Chatbot for EV Charging Management

In this section, we present ReChat, a *task-based* chatbot designed to enhance EV charging management through natural language interaction. The *task-based* approach implies that ReChat focuses on carrying out specific tasks, such as checking the charger's status, scheduling charges or adjusting the charging intensity, through a predefined intent-to-action flow that remains robust to variation in user requests [33], leveraging recent advances in transformer-based NLP (fine-tuned BERT encoders) for robust intent classification and safe action dispatch.

We describe the functionalities offered to users to optimize the use of their charging points. Additionally, we detail the technical aspects of the solution, including the system architecture and the technologies employed in its development. Finally, we explain how users can access and utilize ReChat to efficiently manage the charging of their EVs, advancing task-based dialogue systems for real-world applications in smart transportation.

3.1 | System Architecture

ReChat is built upon a modular and scalable architecture that integrates various technologies to optimize EV charging management. The system's design focuses on ensuring accessibility, security and ease of use, offering a personalized user experience through natural language processing. Moreover, we have ensured that the modules are easily replaceable or expandable to allow the incorporation of new models and technologies in the future, supporting adaptability to diverse linguistic contexts.

The architecture of ReChat is based on the instant messaging platform Telegram, selected for its broad adoption and support for bot creation. Telegram allows users to interact with ReChat through a familiar chat interface without the need to install additional applications.

The system is composed of several key modules, each playing an important role in enabling seamless and efficient charging management. The user interface (UI) is provided through Telegram, where users send queries and receive responses in natural language, facilitating user interaction with the charging points.

Figure 1 illustrates the general architecture of the system. Central to ReChat's functionality is the NLP module, responsible for interpreting user inputs and determining relevant intents and parameters. After evaluating multiple approaches for implementing this module, we opted for a BERT-based intent classifier as the best overall trade-off for our deployment constraints. While *Voiceflow* w/ GPT achieved higher intent-classification $F1$ in our evaluation, BERT provides stronger control over execution, on-prem deployment, predictable latency and an auditable intent-to-action layer, which are desirable properties for charging-control operations.

Although *Voiceflow* w/ GPT achieved higher $F1$ in our experiments, we selected the BERT-based pipeline because it offers a more controllable execution regime for charging-control actions. In particular, it enables on-premise inference, predictable latency, and a clear separation between intent prediction and deterministic backend execution, reducing reliance on third-party services while supporting auditing and safety policies.

The backend and services, implemented in Node.js, handle the business logic and communication with the charging devices through a dedicated API. This ensures that all operations are performed securely and efficiently. The Telegraf library facilitates interaction with the Telegram API, enabling real-time communication between the chatbot and users.

Beyond the generic chat pipeline, ReChat includes an EV-specific orchestration layer. This layer enforces device and safety policies (e.g., charger state-machine guards and min/max current limits), and normalizes vendor/OCPP APIs into consistent endpoints. It also validates parameters and units (A/kW and locale), and logs operations and outcomes for auditability. These checks ensure that intents such as `intensity`, `pause` or `schedules` translate into safe and deterministic back-end requests. Algorithm 1 formalizes the execution pipeline of this orchestration layer, detailing the sequential validation stages that ensure safe intent-to-action dispatch.

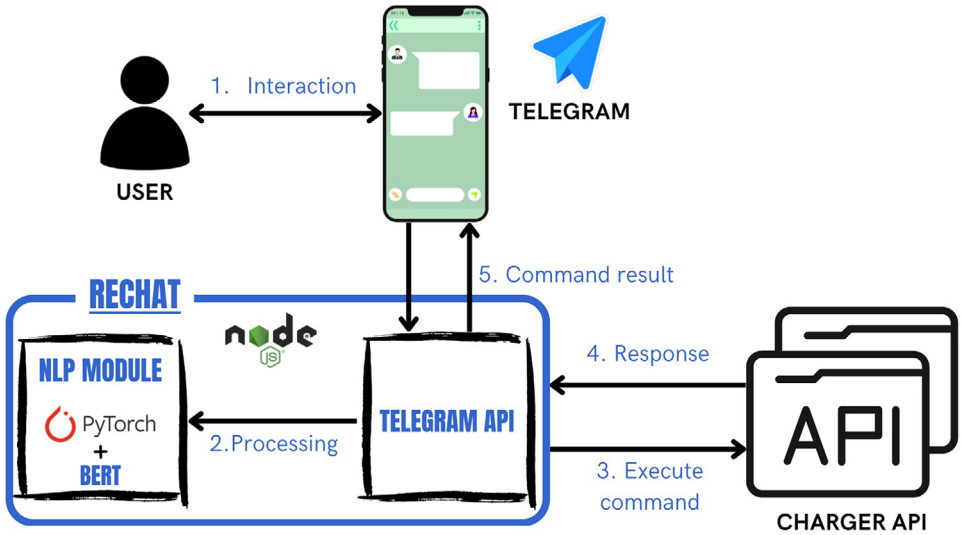


FIGURE 1 | ReChat system architecture.

As shown in Algorithm 1, the orchestration layer implements a guarded execution pipeline with five key validation stages. First, intent classification and slot extraction produce a confidence score; low-confidence predictions trigger clarification prompts rather than proceeding with potentially incorrect actions. Second, authentication ensures that only authorized users can dispatch backend operations, with requests bound to the user–device association established by the charging API. Third, state-machine validation checks whether the requested action is admissible in the current charger state (e.g., `resume` requires a paused session). Fourth, parameter bounds checking validates numeric inputs (e.g., `intensity` values) against device-specific min/max limits. Finally, if all validations pass, the intent+slots are mapped deterministically to a single idempotent backend call, with the result logged for auditing. This multi-stage validation ensures that no backend operation executes under uncertainty, providing practical robustness against equipment faults, transient connectivity issues, and ambiguous natural-language requests.

The interaction flow in the system is as follows:

1. The user sends a query through Telegram.
2. The message is received by the ReChat bot and sent to the NLP module for intent classification.
3. Once the intent is determined, the system executes the corresponding action by communicating with the charging device via the API.
4. The result of the action is returned to the user through a message in Telegram.

According to this, user text is tokenized and encoded by the fine-tuned BERT encoder, and a linear classification head produces an intent distribution. Lightweight slot extraction (regex/templates per intent) retrieves parameters when present (e.g., target current in `intensity`). The intent+slots are validated by the orchestration layer and executed via a deterministic policy that maps them to a single back-end call (see Table 2). Finally, the result is summarized back to the user. This task-to-action mapping is fixed and auditable, eliminating the need for free-form generation at execution time.

Depending on the deployment configuration, intent classification is performed either by language-specific mBERT (one model per language) or by a single pooled multilingual mBERT model trained on the union of languages, enabling a direct comparison between per-language specialization and pooled multilingual training within the same execution pipeline.

To coordinate intent classification with parameter extraction, ReChat associates each intent with a minimal slot schema (required vs. optional parameters, expected type/unit and admissible ranges). After the intent is predicted, the slot extractor applies intent-specific regex/templates to retrieve parameters (e.g., a target current for `intensity`). The orchestration layer then validates the combined intent+slots by checking (i) the presence of required parameters, (ii) type and unit consistency and (iii) domain constraints such as charger state and min/max allowed setpoints. If a required parameter is missing or is considered invalid, ReChat does not issue any backend request. Instead, it asks a clarification question to obtain a valid value (e.g., requesting an intensity within the allowed bounds). Likewise, when intent confidence is low or when multiple intents are competing, the system asks the user to rephrase or confirm the intended action before proceeding.

Given that ReChat can trigger control operations on charging equipment, the system follows a safe-by-default execution policy. Backend operations are only dispatched for authenticated users and are bound to the user–device association established by the charging API. In addition, the orchestration layer enforces safety guards (e.g., state-machine checks and min/max limits) and rejects malformed or out-of-range parameters to prevent misoperations. All executed actions and outcomes are logged for auditing and troubleshooting, and deployments can be configured to require explicit user confirmation for high-impact operations (e.g., disabling a charger or changing the current setpoint).

ALGORITHM 1 | Orchestration layer: Safe intent-to-action dispatch.

Require: User message m, Intent classifier C, Charging API A
Ensure: Backend response r or clarification prompt p

$\langle i, conf, slots\rangle \leftarrow C(m)$ ▷ Classify intent and extract slots
if $conf < \theta_{\min}$ **then**
  return $p \leftarrow$ "Could you rephrase?"
end if
if ¬ValidSlots$(i, slots)$ **then**
  return $p \leftarrow$ "Missing parameter"
end if
$\langle auth, device\rangle \leftarrow$ Auth()
if $\neg auth$ **then**
  return $p \leftarrow$ "Auth required"
end if
$state \leftarrow A$.GetState$(device)$
if ¬StateMachineOK$(i, state)$ **or** ¬BoundsOK$(i, slots)$ **then**
  return $p \leftarrow$ "Not allowed: [reason]"
end if
$action \leftarrow$ MapToAPI$(i, slots)$
$r \leftarrow A$.Exec$(action, device)$
if r.isError() **then**
  Log$(i, action, r)$
  return $p \leftarrow$ "Failed: [recovery]"
end if
Log$(i, action, r)$
return r

Because ReChat issues control actions on physical charging equipment, the backend explicitly handles abnormal and failure scenarios, thereby avoiding the execution of operations under uncertainty. At runtime, every request follows a guarded pipeline: the bot first validates authentication and the user–device binding, then checks the charger state and parameter bounds, and only then dispatches a single idempotent backend call. If the charging API reports a device fault (e.g., unavailable connector, session error or rejected setpoint), ReChat converts the backend error code into a user-facing explanation and suggests the safest next step (e.g., re-checking status, selecting another device or retrying later), while logging the failure for troubleshooting.

If a network interruption or timeout occurs, the request is not retried blindly; instead, the system returns an explicit 'operation not confirmed' message and asks the user to query the current status before attempting the action again, preventing duplicate or inconsistent state changes. For ambiguous user commands or underspecified control verbs (e.g., 'stop charging' without clarifying whether the target is the session or the charger state), ReChat does not call any backend endpoint and triggers a clarification question to disambiguate the intended action; similarly, when the intent confidence is low, or when the top-2 intents fall into a known confusion cluster, the system requests confirmation or rephrasing. This failure-aware, confirmation-first behaviour is consistent with the deterministic intent-to-action policy of ReChat, providing practical robustness against equipment faults, transient connectivity issues, and ambiguous natural-language requests, while keeping an auditable trace of executed operations and their outcomes.

This modular and scalable architecture ensures real-time interaction and provides a seamless user experience. It also facilitates the integration of new NLP models or services as technology evolves, ensuring the system's continuity and security. Moreover, this approach allows the system to adapt to emerging user needs and technological advancements, maintaining its effectiveness as multilingual task-based dialogue systems evolve.

3.2 | Functionalities of ReChat

ReChat is an intelligent chatbot that enables EV users to manage and optimize their charging devices intuitively and effectively. Through NLP, ReChat offers an interface that allows interaction with charging systems using natural language, thereby improving accessibility and user experience. The main functionalities offered by ReChat, corresponding to the different intents predicted by the NLP module, are as follows:

- Checking current consumption: ReChat allows users to check real-time energy consumption, providing critical data for monitoring the charger's performance and energy efficiency.
- Checking charger status: Users can ascertain the current status of the charger, obtaining information on whether the session is active/paused and whether the charger is enabled/disabled.
- Disabling or enabling the charger: ReChat offers users the ability to fully enable or disable the charger.
- Requesting help: Users can ask ReChat for help with handling the chatbot; ReChat will indicate the different functionalities available and provide prompt examples for each.
- Adjusting charging intensity: ReChat exposes the charging-current setpoint as an optional advanced control. This intent is designed for users who already customize charging current in the charger vendor application, and who understand the implications of current-setpoint adjustments (e.g., reducing current to stay within local breaker limits or to implement manual load balancing), and who prefer to issue the same command via natural language. Charging operations such as starting, pausing, resuming, or monitoring do not require adjusting this parameter; the default current setpoint configured in the charger backend is used automatically. For most users, this parameter is not required, and it is therefore not part of the typical interaction flow. In the current prototype, if the user provides a numeric value, ReChat forwards it to the backend action associated with `intensity`; value admissibility (e.g., device-specific limits) is enforced by the charging backend/API.
- Changing language: Recognizing user diversity, ReChat incorporates a language change function through natural language

TABLE 2 | Mapping of user demands to intents and backend actions (examples).

Demand	Intent(s)	Backend/API action(s)
Monitoring	`actual_state`, `actual_intake`, `statistics`	`GET /status`, `GET /metrics/current`, `GET /charges/recent`
Control	`pause`, `resume`, `enable`, `disable`	`POST /charging/pause`, `POST /charging/resume`, `PATCH /device/enable`, `PATCH /device/disable`
Configuration	`intensity`, `list`, `schedules`	`PATCH /charging/intensity`, `GET /devices`, `GET /schedules`
Support	`help`, `token`, `language`	`GET /help`, `PATCH /auth/token`, `PATCH /session/language`

commands, supporting multilingual interaction in diverse linguistic contexts inspired by LLM capabilities.

- Listing devices: ReChat provides a list of the different chargers associated with the user.
- Operational control over charging: ReChat offers users the ability to pause and resume charging as needed.
- Checking charging schedules: Users can review the different charging schedules they have configured.
- Statistics of recent charges: ReChat provides a summary of the latest charges performed, including details such as duration, energy consumed and an estimated associated cost.
- Changing the authentication token: ReChat allows users to change their authentication token, which is necessary for communication with the charging devices.

These functionalities are designed to offer a comprehensive set of user-facing operations, making EV charging management accessible and efficient. The core monitoring and control operations (checking status, consumption, and statistics; pausing, resuming, enabling and disabling the charger; listing devices; and requesting help) are accessible to all users through straightforward natural-language commands, without requiring technical knowledge of charging parameters. Advanced configuration options (adjusting charging intensity, managing schedules) are provided for users who already interact with these settings in vendor applications and prefer a natural-language alternative.

Beyond the input modality change, ReChat offers several concrete advantages over vendor-specific mobile applications. First, *multi-vendor normalization*: ReChat provides a unified interface for chargers from different manufacturers (e.g., Tesla, Wallbox, ABB), eliminating the need to install and learn separate applications for each device; this is particularly valuable in multi-charger households, fleet deployments, or shared parking facilities where different equipment coexists. Second, *integration with automation ecosystems*: ReChat can be embedded in smart-home or fleet-management workflows, enabling coordinated control with other IoT devices and energy-management systems (e.g., pausing charging when solar production drops, or resuming when dynamic tariffs are favourable); vendor applications typically operate in isolation and do not expose programmable interfaces for third-party automation. Third, *accessibility*: natural-language and voice-compatible interfaces reduce interaction barriers for users with visual or motor impairments, whereas vendor touchscreen-centric UIs can be challenging for these user groups. Fourth, *lower interaction friction for routine operations*: common tasks such as checking status or pausing a session require fewer navigation steps than multi-screen vendor UIs, which often nest basic controls under configuration menus. Fifth, *consistent multilingual support*: vendor applications often provide incomplete or inconsistent translations, whereas ReChat's architecture ensures uniform language coverage across all supported intents and backend operations. Finally, *auditability in shared or fleet scenarios*: ReChat's logged intent-to-action execution enables transparent tracking of who issued which command and when, facilitating accountability and troubleshooting in multi-user environments where vendor apps may lack fine-grained access logs.

To make the operational scope explicit, the supported intents were grouped into four demand categories: Monitoring, control, configuration and support, and mapped to concrete backend actions. Table 2 summarizes this mapping and illustrates representative user prompts per language. The Telegram frontend was connected to the backend (Node.js) and the charger API, enabling end-to-end execution (e.g., reading status/metrics, toggling enable/disable, adjusting current intensity, pausing/resuming sessions and retrieving recent-charge statistics). The scenarios below exemplify typical interactions:

- *Control*: 'Pause charging now' / 'Pausa la carga ya': intent `pause` → API call `POST /charging/pause`.
- *Configuration*: 'Set intensity to 12A' / 'Imposta l'intensità a 12A': intent `intensity` → `PATCH /charging/intensity`.
- *Monitoring*: 'What's the current consumption?' / 'Qual é o consumo atual?': intent `actual_intake` → `GET /metrics/current`.
- *Support*: 'Help' / 'Ayuda': intent `help` → usage hints and examples.

Importantly, `intensity` targets a minority of advanced users who already understand and adjust the charging current through the vendor application. The majority of users can effectively manage their charging sessions using only the core monitoring and control intents (checking status and consumption, pausing

and resuming sessions, enabling and disabling the charger), which require no technical knowledge of electrical parameters or charger configuration. In typical usage cases, most interactions focus on these essential intents rather than advanced configuration such as intensity or schedules. Therefore, ReChat does not assume that current-setpoint adjustment is a mainstream user need; it is included to provide a natural-language alternative for an existing advanced setting that a subset of users already manage through vendor UIs.

These capabilities address typical user needs in EV charging management, including status checks, operational control, configuration and assistance. They demonstrate practical feasibility by mapping each intent to an executable backend operation.

3.3 | Implementation and Resources

The implementation of the intent classifier module involved developing and training transformer-based intent classifiers for multilingual usage. We evaluated two configurations within the same architecture: (i) Language-specific mBERT (one model per language, trained only on that language's utterances) and (ii) pooled multilingual mBERT (a single model trained on the union of all languages). This design allows controlling the deployment trade-off between per-language specialization (higher accuracy under language-specific distributions) and operational simplicity (a single model for all languages).

The development of ReChat also incorporated standard technologies. The system is based on *Node.js*, a server platform that allows handling multiple simultaneous connections, essential for processing user requests in real-time. Interaction with Telegram is managed through the *Telegraf* library, which facilitates the creation of bots and communication with the Telegram API.

For the intent classifier module, a customized model based on *BERT* was developed using the deep learning framework *PyTorch*. This model enables understanding the full context of user queries, improving accuracy in interpretation. To expose the model as a service and to allow communication with ReChat's backend, *Flask*, a Python microframework, was used.

These technologies work together to provide an efficient and scalable user experience, allowing ReChat to handle complex natural language interactions and offer advanced functionalities in electric vehicle charging management.

The datasets used for training and evaluating the NLP models will be available for research purposes upon request to the corresponding author. This will allow other researchers to replicate experiments, analyse the implementation in detail, and contribute to the tool's development. To access the data, please contact the main author of the article through the email address provided in the contact information section.

Users interested in testing ReChat can interact with the chatbot via Telegram. The bot's username on Telegram is @ReChat_evbot. Readers are encouraged to experiment with the chatbot and provide feedback for future improvements.

4 | Methodology

In this section, we describe the methodology followed to implement ReChat, covering dataset construction, model configurations, training, and evaluation of the NLP module. We detail the datasets created, the configuration of each approach, and the training and selection protocol used to support intent classification in a task-based dialogue system.

4.1 | Training and Evaluation Datasets

Since there were no existing datasets that met the specific requirements of the project, we proceeded to create our own dataset. This dataset includes the intents/functionalities that users can express when interacting with the chatbot, including actions such as checking the charger's status, starting or pausing the charging process, adjusting the intensity and changing the language, among others.

For each intent, 20 phrases were generated in six languages: Spanish, English, German, French, Italian and Portuguese.

4.1.1 | Dataset Construction, Annotation and Quality Control

The training corpus was built from scratch because we found no public dataset covering EV charging control intents with multilingual utterances and an action-oriented ontology. We followed a reproducible construction protocol: first, we defined the intent inventory directly from the executable capabilities exposed by the charging backend (Table 2). Each intent was associated with a short natural-language description, a canonical backend action, and a minimal slot schema (required/optional parameters). This slot schema was also used to write intent-level labelling rules that reduce ambiguity in semantically adjacent classes by prioritizing the *action target* as the main discriminator; for example, `pause/resume` refer to the charging session, whereas `disable/enable` refer to the charger availability state.

For each intent, we generated 20 short user utterances in each of the six languages (Spanish, English, German, French, Italian and Portuguese), resulting in 1560 instances. Utterances were designed to cover typical linguistic variability while remaining label-preserving, including imperative and interrogative forms (e.g., 'pause charging' vs. 'can you pause the charge?'), polite versus direct requests, synonym substitutions and word-order variation. To minimize bias and keep the process reproducible, we adopted a controlled template-and-paraphrase workflow: for each intent, we drafted a small set of seed templates in English describing the same action in different registers, and we manually rewrote them into additional paraphrases until reaching the target count while avoiding near-duplicates. For non-English languages, utterances were produced by translating and rewriting the intent-preserving seeds into the target language, explicitly introducing language-specific phrasing and morphology (e.g., clitics, inflection and compounding) rather than relying on literal translations.

Each utterance was assigned exactly one intent label according to the labeling rules above. Labels were produced during dataset creation (i.e., utterances were generated to match a target intent) and then verified in a second pass to ensure that the surface form still matched the intended action. For utterances containing numeric values or device references, we additionally checked that the text was compatible with the intent slot schema (e.g., that `intensity` includes a plausible numeric current value) and that no intent identifiers or backend keywords leaked into the utterance text.

Before training, we applied a lightweight quality-control pipeline: duplicates and near-duplicates were screened within each intent-language subset (manual inspection aided by string similarity), utterances were reviewed for ambiguity in known confusion clusters (e.g., generic 'stop charging' requests that do not specify whether the session should be paused or the charger disabled), and basic normalization checks were applied (punctuation and formatting consistency). Ambiguous utterances that could reasonably map to multiple intents were rewritten to include an explicit target (e.g., 'pause the current session' vs. 'disable the charger'), preserving the deterministic intent-to-action mapping required by ReChat.

For reproducibility, the dataset is stored in a simple tabular format with columns `language`, `utterance`, `intent` and (when applicable) `slots`. In addition, the intent definitions (descriptions and slot schemas) are maintained in a separate machine-readable specification that can be used both to regenerate seed templates and to validate label consistency. The dataset and specifications will be shared for research purposes upon request, as stated in Section 3.

This study uses six languages for three pragmatic reasons: (i) to broaden the potential user base by covering languages spoken by a substantial share of the global population, (ii) to achieve typological variety within a fixed annotation budget (Romance and Germanic families), and (iii) to leverage a single, widely adopted multilingual transformer checkpoint (mBERT) that supports all six languages with a unified tokenizer, enabling controlled comparisons between language-specific mBERT and pooled multilingual mBERT within the same modelling backbone.

The corpus comprises 1560 utterances, 20 per intent and per language. Each instance consists of a user phrase paired with its intent label. This size is sufficient for a transfer learning pilot, as it preserves per-intent balance, constrains annotation cost and yields stable validation signals for hyperparameter selection. Accordingly, we adopted a stratified 80/20 training and validation split by language and intent.

The creation of the dataset allowed for a complete representation of user interactions in multiple languages, ensuring that the chatbot can handle a wide range of expressions. This approach enhances the model's ability to generalize across different linguistic contexts and user intents, providing a foundation for multilingual conversational reasoning in task-based dialogue systems.

To evaluate the trained models, three evaluation datasets were generated using different advanced language models: GPT-4o, Gemini 1.5 Pro and Claude 3.5 Sonnet. Through these models, various prompts were generated, resulting in a broad set different from the training set. This use of LLMs for evaluation ensured a diverse and challenging test of the chatbot's intent classification capabilities, aligning with the goal of advancing real-world applications. This allows for a precise and fair evaluation of the models' performance.

4.2 | Configuration and Training Processes

As seen in Section 2, there are different technologies to implement the intent classifier module in task-based chatbots. Therefore, different training processes were carried out according to the approach used, with the aim of determining the most appropriate technology for ReChat.

4.2.1 | Voiceflow w/ GPT

The first approach uses *Voiceflow* w/ GPT. In this case, the user's message is sent to the GPT model, which returns the intent as a specific command. This approach leverages the model's ability to understand and process natural language efficiently, facilitating the creation of fluid and natural dialogues. By using an LLM like GPT, this method enables ReChat to interpret complex user inputs in a task-based dialogue system.

Since this is a prompting-based configuration rather than supervised training, there is no dataset-driven fine-tuning step. Instead, we controlled generation to maximize determinism in intent routing by setting the sampling temperature to 0.0 and limiting the output budget (maximum length of 512 tokens), yielding consistent command outputs under our evaluation protocol.

4.2.2 | Voiceflow w/ NLP Module

The second approach involves manually defining intents and prompts in *Voiceflow*. The system is trained with a set of representative phrases for each intent, covering the different supported languages. The training was conducted automatically through *Voiceflow*, processing the utterances to effectively classify user inputs. This method involves a detailed process of analysing and defining intents, ensuring complete coverage of the users' communicative needs without relying on external models.

4.2.3 | mBERT

The third approach involves the development and training of an mBERT-based model for multilingual intent classification. Using *PyTorch* and *Flask*, a model capable of accurately interpreting user queries in multiple languages was fine-tuned. Two variants were explored: language-specific mBERT (one model per language) and pooled multilingual mBERT (a single model trained on the union of all languages). Although mBERT is not an LLM, its transformer-based architecture shares foundational principles with LLMs, making it suitable for precise intent classification in task-based dialogue systems, particularly in multilingual contexts.

All mBERT classifiers in this study were initialized from the same multilingual pre-trained checkpoint, `bert-base-multilingual-cased` (mBERT), and fine-tuned with a linear classification head for intent prediction. Using a single multilingual checkpoint across experiments ensures that the comparison between language-specific mBERT and pooled multilingual mBERT isolates the effect of data partitioning (per-language vs. union of languages) rather than differences in pre-training corpora or tokenization.

To keep the comparison focused on the effect of data partitioning, we kept the encoder architecture, tokenizer, and preprocessing pipeline fixed across all mBERT experiments. Given the limited corpus size, this choice provides a controlled and reproducible baseline for multilingual intent classification.

For mBERT fine-tuning, we used the 1560-utterance corpus with a stratified 80/20 split for training and validation by language and intent. Models were optimized for multi-class intent prediction and monitored on the validation split to select the best configuration while mitigating overfitting under limited data.

Hyperparameters were selected via a grid search on the validation split. We explored batch sizes of 16, 32 and 64; learning rates of 1×10^{-5}, 3×10^{-5} and 5×10^{-5}; training epochs of 20, 30 and 40; and maximum sequence lengths of 10, 15 and 20 tokens. For each model (language-specific or pooled), we retained the configuration that maximized validation performance (macro-$F1$ in the comparative evaluation), and we then reported test performance on the LLM-generated evaluation sets using the selected hyperparameters.

The selection criterion was strictly based on maximizing macro-$F1$ score on the validation split; no additional weighting was applied to loss magnitude or training stability. This single-metric approach ensures reproducibility and provides a clear benchmark for comparing language-specific versus pooled multilingual configurations under identical evaluation protocols.

5 | Results

This section reports and discusses the results obtained from evaluating the different approaches implemented for the NLP module of ReChat. We present (i) the evaluation of the *Voiceflow* implementations, (ii) the training behaviour of the mBERT models, (iii) the evaluation performance of the mBERT models, (iv) an end-to-end comparison of all approaches and (v) complementary analyses on computational cost (inference latency) and error patterns.

5.1 | Evaluation of *Voiceflow* Implementations

Regarding the implementations in *Voiceflow*, two different configurations have been evaluated: *Voiceflow* w/ GPT, and *Voiceflow* w/ NLP module.

Table 3 summarizes the per-language $F1$-score obtained for both configurations, enabling a direct comparison of multilingual intent-classification performance under the same evaluation protocol.

TABLE 3 | $F1$-score performance metrics for *Voiceflow* implementations by language.

Language	*Voiceflow* w/ GPT	*Voiceflow* w/ NLP module
English	92.72	81.75
Spanish	94.63	52.28
German	89.45	55.31
French	91.87	53.22
Italian	90.12	50.78
Portuguese	92.34	55.49
All (macro avg)	**91.86**	**58.14**

As shown in Table 3, *Voiceflow* w/ GPT attains per-language $F1$-scores in the range 89.45%–94.63% (macro average 91.86). In contrast, *Voiceflow* w/ NLP Module yields a macro average of 58.14, with English at 81.75 and the remaining languages between 50.78 and 55.49.

This gap is consistent with: (i) the stronger multilingual generalization of LLM-based routing under prompt supervision and (ii) the more limited cross-lingual robustness of the platform's built-in intent classifier when trained with a small set of manually declared examples. In addition, the linguistic variability of Romance and Germanic languages (inflection, clitics, compounding, and word-order differences) can amplify errors when the underlying embeddings or intent features are not optimized for multilingual coverage in this setup.

On the other hand, *Voiceflow* w/ NLP module, which is based on manual definition of intents, shows notably lower performance in languages other than English. Despite efforts to train the model with specific intents for each language, the results indicate that *Voiceflow*, under our configuration and evaluation protocol, exhibits substantially higher robustness in English than in the remaining languages. This is evidenced by significantly lower $F1$-scores in Spanish, German, French, Italian and Portuguese, reflecting the platform's limitations in effectively handling non-English languages across multiple languages.

These results reflect that *Voiceflow* w/ NLP module presents inconsistencies and limitations, particularly in a multilingual environment. Intents that require greater contextual understanding, such as `statistics` and `actual_intake`, show significant deficiencies, suggesting that improvements are needed in the module's ability to handle variations and complexities in linguistic contexts.

In summary, *Voiceflow* w/ GPT positions itself as a more viable and effective option for intent classification in multiple languages, leveraging LLM capabilities, while *Voiceflow* w/ NLP module requires significant improvements to reach comparable performance levels.

TABLE 4 | Hyperparameters used in training the mBERT models.

Model	Learning rate	Batch size	Epochs	Sequence length
Pooled multilingual mBERT	3×10^{-5}	16	30	20
Language-specific mBERT (EN)	5×10^{-5}	16	20	10
Language-specific mBERT (ES)	1×10^{-5}	16	30	20
Language-specific mBERT (DE)	1×10^{-5}	16	40	20
Language-specific mBERT (FR)	1×10^{-5}	16	40	20
Language-specific mBERT (IT)	3×10^{-5}	16	30	20
Language-specific mBERT (PT)	1×10^{-5}	16	30	20

5.2 | Training Performance of mBERT Models

To evaluate intent-classification capabilities in multilingual environments, several mBERT models adapted to ReChat's needs were trained. On one hand, a Pooled Multilingual mBERT variant was trained to cover several languages simultaneously. On the other hand, language-specific mBERT models were developed for each of the languages of interest (Spanish, English, French, Italian, Portuguese, and German).

The selection of these two approaches offers a more comprehensive comparative analysis, ranging from performance in each language separately to versatility in environments where multiple languages are mixed. Table 4 lists the hyperparameters used by each model.

All hyperparameters (i.e., learning rate, batch size, epochs and maximum sequence length) were selected per language via grid search on the validation split, aiming to balance convergence speed, stability and overfitting. Languages with longer or more compositional utterances and higher morphological variability benefited from longer sequences and, in some cases, additional epochs (e.g., German and French at 40 epochs and length 20), whereas English reached a stable validation plateau with fewer epochs and a shorter sequence length (20 and 10, respectively). Learning rates were chosen to reduce validation-loss volatility and overshooting (smaller rates for slower-converging languages), and the batch size converged to 16 across languages because larger values did not improve validation metrics and increased memory consumption.

Figure 2 illustrates representative training dynamics (loss and accuracy) for language-specific mBERT (EN), language-specific mBERT (ES), and pooled multilingual mBERT, which reflect the overall trends observed across languages.

As can be observed, all models present a similar trend in terms of loss decrease and accuracy increase over the epochs. This indicates that, in general, the training process has been successful in all cases.

However, there are some notable differences between the models. The language-specific mBERT (EN) model shows faster convergence, reaching high accuracy in fewer epochs than the language-specific mBERT (ES) and pooled multilingual mBERT models. This may be partially explained by differences in how quickly the model adapts to the English subset under limited data, possibly influenced by the representation of English in pretraining, and by the linguistic variability of the other languages.

Across runs, we did not observe clear symptoms of severe overfitting in the learning curves (e.g., a sustained divergence between training and validation loss), although the small size of the corpus limits how confidently overfitting can be diagnosed from curves alone.

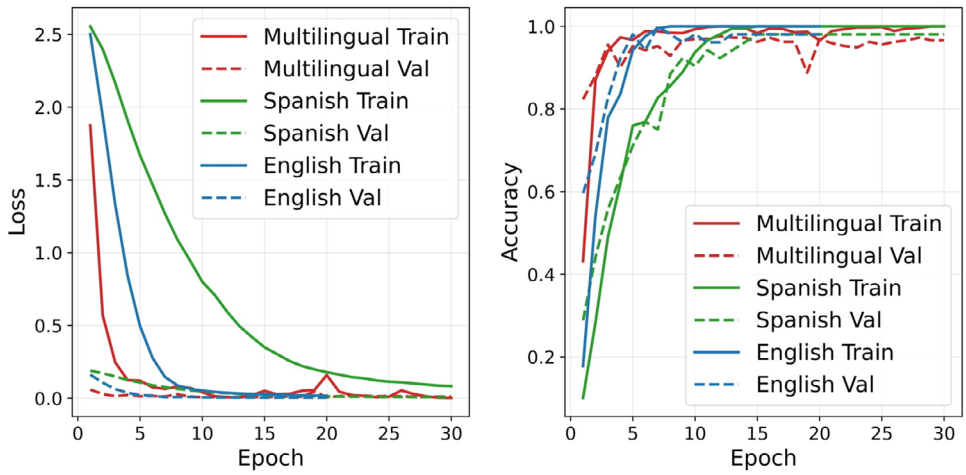
It is important to highlight that the convergence of the pooled multilingual mBERT model is slower compared to the language-specific mBERT models. This is because the model must learn to generalize from data in several languages, which represents a greater challenge. Despite this difficulty, the model manages to achieve competitive training accuracy, demonstrating its ability to classify intents in a multilingual environment. In validation, the pooled model tended to be slightly less stable than the language-specific models, which is consistent with the harder generalization problem under the same data budget.

5.3 | Evaluation of mBERT Models

To verify the behaviour of the models in real interaction situations, the mBERT models have been evaluated using test datasets generated by different advanced language models: GPT-4o, Gemini 1.5 Pro and Claude 3.5 Sonnet. These evaluation datasets contain prompts in the six languages, which are different from those generated for the training set, thereby enabling an assessment of the effectiveness of the models in handling diverse and challenging inputs.

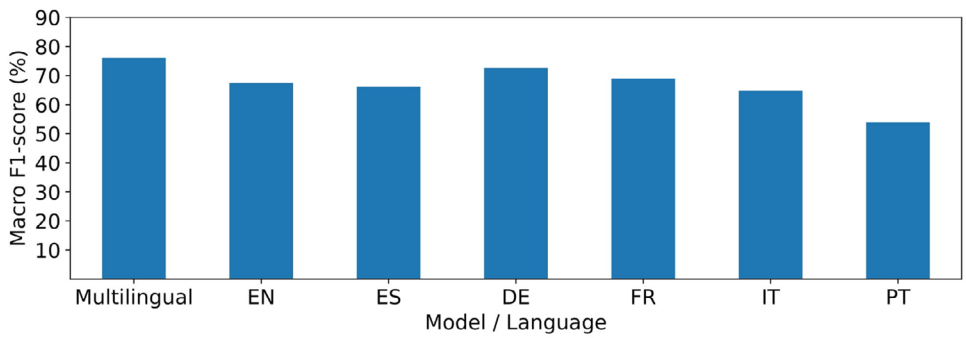
Figure 3 reports the macro $F1$-score obtained by each mBERT model on the three LLM-generated evaluation sets, highlighting the impact of prompt distribution and language on intent-classification performance.

As shown in Figure 3, $F1$-scores vary across the GPT-4o, Gemini 1.5 Pro, and Claude 3.5 Sonnet evaluation sets due to distributional shifts in the generated prompts (imperative vs. colloquial registers, rare synonyms, code-mixing, punctuation and utterance length), which modify class separability and amplify confusions among semantically close intents, especially in languages with richer inflection or productive compounding.

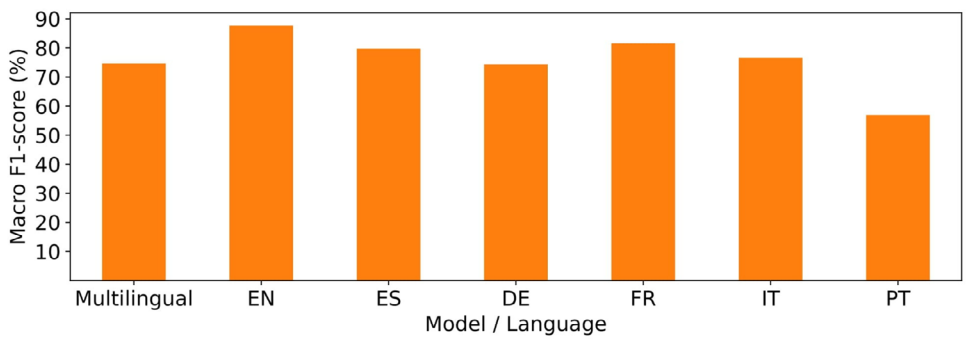


(a) Loss history (b) Accuracy history

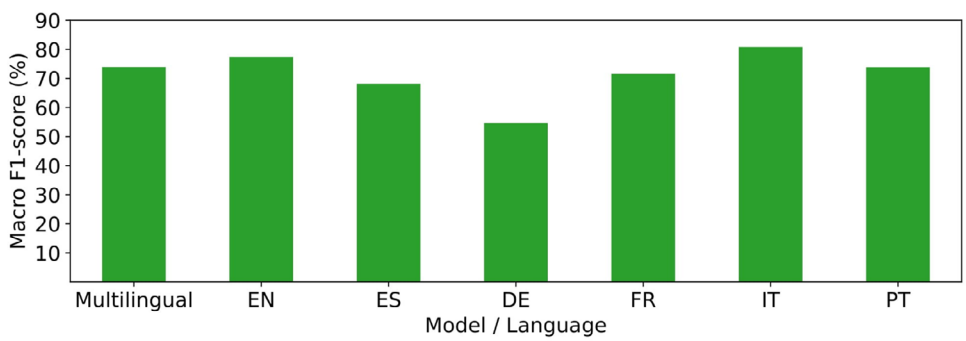
FIGURE 2 | Loss and accuracy curves during the training of the mBERT models: (a) loss history and (b) accuracy history.



(a) GPT-4o



(b) Gemini 1.5 Pro



(c) Claude 3.5 Sonnet

FIGURE 3 | Macro $F1$-score of mBERT intent classifiers on the three LLM-generated evaluation sets.

TABLE 5 | Comparative performance ($F1$-score) of different NLP implementations by language.

Language	*Voiceflow* w/ GPT	*Voiceflow* w/ NLP module	Language-specific mBERT	Pooled multilingual mBERT
English	92.72	81.75	78.861	78.331
Spanish	94.63	52.28	72.853	78.393
German	89.45	55.31	69.175	67.990
French	91.87	53.22	77.086	73.217
Italian	90.12	50.78	74.916	77.259
Portuguese	92.34	55.49	64.652	74.880
All (macro avg)	**91.86**	**58.14**	**72.92**	**75.01**

Overall, both mBERT configurations achieve competitive intent-classification performance under the same backbone and preprocessing pipeline. In our evaluation, pooled multilingual mBERT attains a slightly higher macro-average across the six languages (Table 5), with clear gains in Spanish, Portuguese and Italian. Language-specific mBERT remains advantageous for some languages (notably English and French, and marginally German), indicating that the relative ordering depends on the target language and the prompt distribution.

The performance differences between language-specific mBERT and pooled multilingual mBERT can be partially explained by cross-lingual transfer effects and linguistic typology. Pooled multilingual mBERT benefits from implicit regularization across languages during training: shared linguistic structures (e.g., similar imperative forms in Romance languages, or common Germanic compounding patterns) enable the model to learn more robust intent representations by pooling evidence across typologically related languages. This is particularly evident for lower-resource configurations within our dataset (e.g., Portuguese and Italian), where the pooled model can leverage patterns from Spanish and French to improve generalization. Conversely, language-specific mBERT can overfit to language-specific surface patterns (e.g., lexical cues or fixed word orders) that do not generalize well to the distributional shifts introduced by LLM-generated evaluation prompts. For English and French, where the training data may contain more consistent phrasing conventions, the language-specific model can exploit these patterns effectively; however, for languages with richer morphological variability (e.g., German declensions, Spanish verb conjugations), the pooled model's cross-lingual smoothing appears to prevent overfitting to spurious correlations in the limited training data.

It is worth noting that the Gemini 1.5 Pro evaluation set yields the highest macro-F1 for the language-specific mBERT models, whereas the GPT-4o set favours pooled multilingual mBERT (Table 7). This confirms that LLM-generated evaluation data can induce distribution shifts that change the relative ordering between models.

While pooled multilingual mBERT simplifies deployment and maintenance (single model for all languages), language-specific mBERT can be preferable when optimizing for a particular language, or when language-level distributions differ substantially. Therefore, the choice between pooled and language-specific training should be driven by deployment constraints and language-specific performance targets.

In summary, both configurations provide strong intent-classification performance under limited data. Pooled multilingual mBERT improves overall macro-average and simplifies deployment, while language-specific mBERT can still be advantageous for selected languages. Consequently, the most appropriate choice depends on whether the deployment prioritizes single-model maintenance or maximum performance in a specific target language.

5.4 | Comparison of Different Implementations

In this section, a comparison of the different approaches implemented for the NLP module of ReChat is conducted to select the most suitable one. The results obtained with *Voiceflow* (*Voiceflow* w/ GPT and *Voiceflow* w/ NLP module) and with mBERT models (language-specific mBERT and pooled multilingual mBERT) are analysed, considering both the precision in intent classification and other relevant factors such as implementation complexity, scalability and computational cost, with a focus on their effectiveness in supporting task-based dialogue systems.

Table 5 compiles the per-language $F1$-score of the evaluated approaches, enabling an end-to-end comparison between rapid-deployment (*Voiceflow*) and supervised (mBERT) intent-classification pipelines. We report macro-$F1$ across intents, computed separately for each language subset.

As shown in Table 5, *Voiceflow* w/ GPT attains the highest $F1$ across all languages (89.45%–94.63%), which is consistent with the different modelling regimes: a generative, cross-lingual LLM with dynamic prompting versus supervised classifiers trained on a 1560-utterance corpus. In addition, LLM-generated evaluation prompts may partially match the stylistic priors of the same model family used for intent routing, which can further benefit the *Voiceflow* w/ GPT configuration.

In contrast, mBERT performance is constrained by the limited size of the supervised corpus and by the difficulty of separating semantically adjacent intents under short, heterogeneous user utterances; nevertheless, it avoids reliance on external services and offers greater control and customizability. We consider

that intent-$F1$ can be increased in the mBERT pipeline by: (i) expanding per-intent diversity via paraphrase/back-translation and code-mix augmentation; (ii) class-balanced sampling and focal loss to mitigate rare-class confusions; (iii) per-language threshold calibration and temperature scaling; (iv) modestly longer sequences and domain-adaptive pretraining on EV text and (v) evaluating stronger multilingual backbones (e.g., XLM-R or mDeBERTa) under the same training protocol.

Among the supervised options, pooled multilingual mBERT provides the best overall trade-off in our experiments: it achieves the highest macro-average across the six languages while requiring only a single on-prem model for deployment and maintenance (see Table 5). Language-specific mBERT remains beneficial for particular languages (e.g., English and French in our evaluation), but it increases operational complexity by requiring one model per language and separate hyperparameter/monitoring workflows.

Finally, *Voiceflow* w/ NLP module shows the lowest performance among all approaches, especially in languages other than English, highlighting its limitations in supporting multilingual inputs.

Considering the requirements of auditable intent-to-action execution and independence from third-party inference services, we select supervised mBERT-based intent classification as the most suitable option for ReChat's NLP module. Within this supervised family, pooled multilingual mBERT is the best overall choice under our data budget and deployment constraints, while language-specific mBERT remains an alternative when maximizing performance for a specific target language is the primary requirement.

Beyond intent-classification accuracy, the choice between LLM-based routing (*Voiceflow* w/ GPT) and supervised mBERT-based classification involves additional engineering trade-offs. First, *uncertainty handling*: mBERT produces calibrated softmax distributions over intents, enabling straightforward confidence thresholding and explicit clarification prompts when the top-1 prediction is ambiguous; in contrast, LLM-based routing can hallucinate plausible-sounding intent labels that do not map to executable backend operations, requiring additional output validation. Second, *out-of-domain robustness*: when users issue commands outside the supported intent set (e.g., general conversational queries or requests unrelated to EV charging), mBERT's closed-set formulation allows deploying a reject-class threshold or an explicit out-of-scope detector, whereas LLM-based routing may attempt to force-fit the input into an existing intent, increasing the risk of unintended actions. Third, *interactive clarification*: ReChat's orchestration layer implements state-aware disambiguation (e.g., distinguishing `pause` from `disable` based on current charger state), which is easier to integrate with a deterministic intent classifier than with a generative LLM where prompt engineering and output parsing introduce additional failure modes. Finally, *auditability and version control*: mBERT-based intent classification allows freezing model checkpoints, versioning training data, and reproducing evaluation results deterministically, whereas LLM API endpoints can change behaviour across releases, complicating long-term maintenance and regulatory compliance in safety-critical deployments.

5.5 | Computational Cost: Inference Latency

To complement the accuracy-based comparison with quantitative evidence of computational cost, we benchmarked inference latency for the mBERT intent classifiers using a fixed batch size and the same preprocessing pipeline as in evaluation. Table 6 reports average latency per batch and per sample for a representative language-specific mBERT model (Spanish) and for pooled multilingual mBERT. Measurements were obtained on a MacBook Pro with Apple M2 Pro (CPU-only inference) and 32 GB RAM.

Overall, the measured per-sample inference time remains within a few milliseconds in our benchmark, which is compatible with interactive chatbot usage. This quantifies the computational cost of the mBERT-based approach and enables a more concrete comparison against rapid-deployment solutions. While the latency is acceptable for a Telegram chatbot, it remains a relevant engineering consideration for resource-constrained deployments (e.g., embedded gateways), motivating future optimization via model distillation, quantization or compact transformer backbones.

5.6 | Error Analysis and Statistical Significance

To better understand the limitations of the intent classifiers, we performed a paired error analysis on the three LLM-generated evaluation sets (GPT-4o, Gemini 1.5 Pro and Claude 3.5 Sonnet). In addition to macro-$F1$, we applied paired statistical tests to assess whether the observed differences between language-specific mBERT and pooled multilingual mBERT are reliable. Specifically, we used: (i) an exact McNemar test on paired correctness outcomes and (ii) a non-parametric bootstrap (2000 resamples) to estimate the distribution of the macro-$F1$ difference (language-specific mBERT minus pooled multilingual mBERT) and its 95% confidence interval (CI). For McNemar, we report n_{01} (language-specific mBERT incorrect, pooled multilingual mBERT correct) and n_{10} (language-specific mBERT correct, pooled multilingual mBERT incorrect).

Overall, the bootstrap analysis indicates no statistically reliable difference between language-specific mBERT and pooled multilingual mBERT on the Claude and Gemini evaluation sets, as the 95% confidence intervals include zero. However, on the GPT-4o set, the bootstrap difference is significantly negative (lang.-spec. minus pooled multi), with a 95% CI entirely below zero ($p = 0.012$), indicating a measurable macro-$F1$ advantage for pooled multilingual mBERT under this evaluation distribution. In contrast, McNemar's test does not reject the null hypothesis in any source, which is expected because it tests paired correctness outcomes rather than macro-$F1$ and can be less sensitive to class-balanced performance shifts.

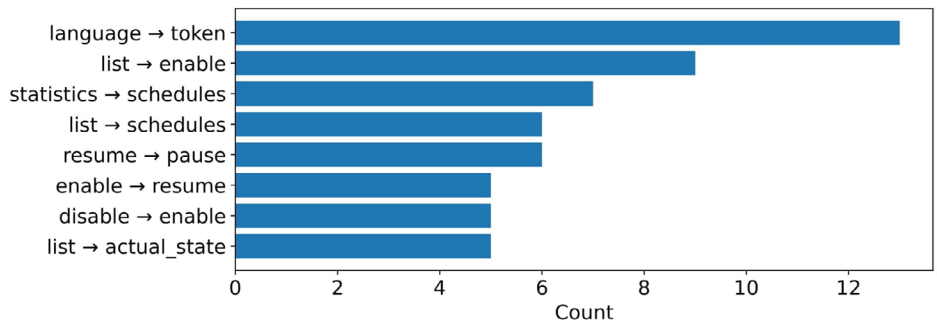
5.6.1 | Most Frequent Confusions and Error Drivers

We further analysed systematic model errors by extracting the most frequent misclassification pairs.

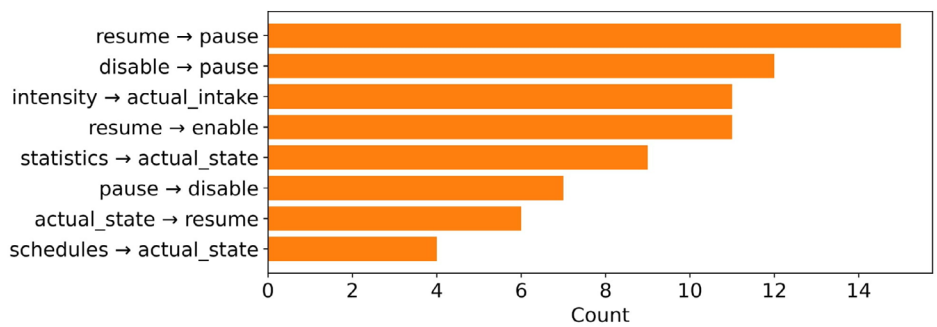
For consistency with the evaluation artefacts, we report intent identifiers exactly as they appear in the prediction logs (e.g., `actual_state` and `statistics`).

TABLE 6 | Inference latency for mBERT intent classification (representative benchmark).

Model	Batch size	Avg time/batch (s)	Avg time/sample (ms)
Language-specific mBERT (ES)	32	0.1405	4.39
Pooled multilingual mBERT	32	0.1494	4.67



(a) Language-specific mBERT



(b) Pooled Multilingual mBERT

FIGURE 4 | Most frequent confusion pairs for language-specific mBERT and pooled multilingual mBERT (counts aggregated over the evaluation sets).

Figure 4 visualizes the dominant confusion patterns for both mBERT settings, highlighting that most errors concentrate on semantically adjacent intents and underspecified user commands.

To make these confusion patterns more concrete beyond counts, we provide illustrative English-only short utterances typical of chat-based EV charging control. For the `pause` versus `disable` cluster, underspecified commands such as 'Stop it', 'Turn it off' or simply 'Pause' may be interpreted either as stopping the current session or making the charger unavailable; a safe mitigation is to ask a clarification like 'Do you want to pause the session or disable the charger?' before dispatching any backend action. For the `resume` versus `enable` cluster, brief prompts such as 'Start again,' 'Turn it back on,' or 'Continue' can refer to resuming a paused session or enabling a disabled charger; again, a practical mitigation is to combine a clarification question with state-aware wording (e.g., 'Is the session paused or is the charger disabled?'). For monitoring intents, `statistics` versus `actual_state` can be confused when users omit temporal markers: 'Status?' or 'Is it charging now?' should map to `actual_state`, while 'How did it go?' or 'What happened last time?' are more consistent with `statistics`; in addition to clarification prompts, adding targeted negative examples and emphasizing time cues ('last,' 'previous,' 'today') helps reduce overlap between these semantically adjacent classes.

The dominant confusions concentrate on semantically adjacent intents and on user utterances that can be underspecified at the surface level. First, session control versus device control is frequently conflated (`pause` vs. `disable`, `resume` vs. `enable`), because many user expressions correspond to a generic goal of 'stopping' or 'starting' charging without explicitly stating whether the target is the charging session or the charger state. Second, monitoring intents can overlap when phrased briefly or in an imperative style (`statistics` vs. `actual_state`), especially when the prompt does not explicitly mention a time span ('last session,' 'today,' 'previous charge') versus instantaneous status ('now,' 'current status'). Third, support/configuration intents can be confused when users request assistance in changing a setting (`help` misread as `token` or `language`) due to shared lexical cues such as 'change,' 'update,' 'how do I' and 'configure.'

These findings motivate two practical mitigations already compatible with ReChat's deterministic execution policy: (i) adding lightweight disambiguation prompts for ambiguous control verbs (e.g., 'Do you want to pause the current session or disable the charger?') and (ii) refining intent-specific templates and negative

TABLE 7 | Macro-$F1$ (%) by evaluation source and paired significance tests (language-specific mBERT vs. pooled multilingual mBERT).

Source	Macro-$F1$ (Lang.-spec.)	Macro-$F1$ (Pooled multi)	n_{01}	n_{10}	McNemar p	Bootstrap diff (Lang.-spec.−Pooled multi) [95% CI], p
Claude (3.5 Sonnet)	73.9	73.9	23	25	0.885	−0.24 [− 8.95, 8.47], 0.938
Gemini (1.5 Pro)	78.0	74.6	18	20	0.871	+3.32 [− 4.31, 10.99], 0.421
GPT-4o	67.1	76.1	27	16	0.126	−9.11 [− 16.01, −2.64], 0.012

examples for semantically close classes (e.g., `statistics` vs. `actual_state`, `pause` vs. `disable`). In addition, deploying a confidence-based clarification policy (already supported in the orchestration layer) can prevent execution when the predicted intent distribution is flat or when the top-2 intents belong to a known confusion cluster.

5.6.2 | Language-Level Variability

Although Figures 3 and 7 aggregate results by evaluation source, language-level inspection shows that the relative behaviour of language-specific mBERT versus pooled multilingual mBERT can differ substantially by language and prompt distribution. For example, in the Claude evaluation set, Spanish shows a higher accuracy for pooled multilingual mBERT than for language-specific mBERT, whereas English on Gemini shows the opposite trend. This reinforces the conclusion drawn from the paired tests: performance differences are sensitive to distributional shifts introduced by the evaluation source and to cross-lingual transfer effects, and therefore should be interpreted with caution under the current corpus size.

6 | Conclusions and Future Work

In this work, we present ReChat, an intelligent multilingual chatbot for electric vehicle charging management. The system architecture, the functionalities it offers, and the technologies used in its development have been described. Additionally, a comparative evaluation of different approaches for implementing the NLP module was conducted, including conversational platforms like *Voiceflow* and transformer-based intent classifiers, covering *Voiceflow* w/ GPT, *Voiceflow* w/ NLP module, language-specific mBERT, and pooled multilingual mBERT, with a focus on their effectiveness in supporting conversational reasoning within task-based dialogue systems.

Our experiments show that LLM-based routing (*Voiceflow* w/ GPT) achieves the highest intent-classification $F1$ across languages in our evaluation. However, for charging-control operations that must be auditable, predictable, and deployable on-prem, supervised intent classifiers provide a more controllable intent-to-action layer and reduce dependencies on third-party inference services.

The evaluation results demonstrate that supervised mBERT-based intent classification offers a favourable balance between accuracy and controllability for ReChat, enabling on-prem deployment and an auditable intent-to-action layer. In our experiments, pooled multilingual mBERT achieves the highest overall macro-average across the six languages, while language-specific mBERT remains beneficial for particular target languages (e.g., English and French). Therefore, pooled multilingual mBERT is the preferred default under our data budget and maintenance constraints, with language-specific mBERT as an alternative when optimizing a specific language is the primary goal.

ReChat's development contributes to the field of vehicular technology by providing an intelligent solution for managing electric vehicle charging. Using a chatbot with advanced NLP capabilities simplifies interactions with charging devices, making essential charging operations (monitoring, pause/resume and enable/disable) accessible to all users through natural language, while optionally exposing advanced configuration parameters for users who already manage these settings, improving the overall user experience and operational efficiency in real-world applications of smart transportation systems.

A key limitation of the current study is the modest size of the training dataset (1560 utterances), which, despite being balanced by intent and language, may not capture long-tail phrasing, typos, code-mixing or regional variants that appear in real deployments. The constraint of 20 utterances per intent per language limits the statistical diversity available for training, potentially reducing the model's ability to generalize to unseen linguistic constructions, dialectal variants, and spontaneous user errors (e.g., typos, autocorrect artifacts, or incomplete commands). This data-size constraint particularly affects the performance on semantically adjacent intent pairs (e.g., `pause` vs. `disable`) where additional training examples with explicit disambiguating cues would strengthen class boundaries. Quantitatively, we estimate that expanding to 50–100 utterances per intent, combined with targeted data augmentation (back-translation, synonym substitution and controlled paraphrasing), could yield macro-$F1$ improvements in the range of 3–8 percentage points, based on preliminary ablation experiments on a held-out subset. In addition, our evaluation relies on test sets generated by large language models, which provide diversity but may not fully reflect the distribution of real user interactions. Finally, we acknowledge that many public charging stations follow a minimal-interaction paradigm (plug-and-charge); therefore, ReChat is especially relevant in scenarios where conversational control is beneficial, such as home/workplace chargers under local power constraints, fleet/depots and tariff- or schedule-aware charging management.

Beyond these limitations, ReChat can have a positive impact on the EV charging ecosystem by offering a natural-language, multilingual interface that makes charging monitoring and control more accessible and auditable. By mapping user intents to deterministic backend operations, the system can increase transparency and user trust, facilitate integration with charging-management backends (including optimization modules such as smart charging or load balancing), and reduce interaction friction for operators and end users (particularly in multilingual environments and IoT settings).

As future work, we propose integrating ReChat with other instant messaging platforms and enhancing its compatibility with energy management systems and smart grids. Additionally, we will explore incorporating new functionalities, such as modifying existing charging schedules or optimizing charging based on dynamic electricity tariffs, further enhancing user interaction.

To improve robustness and generalization, we plan to expand the training corpus and apply label-preserving data augmentation techniques (e.g., controlled paraphrasing and back-translation, noise injection and targeted hard negatives for semantically close intents). We also plan to complement LLM-generated evaluation with real-user testing and log-based error analysis, in order to better approximate practical usage and capture natural variability in user expressions.

From a multilingual modelling perspective, future work will also explore: (i) stronger multilingual pretraining backbones for pooled multilingual mBERT-style training, which could further exploit the positive cross-lingual transfer effects observed in our pooled configuration; models, such as XLM-R or mDeBERTa offer larger multilingual pretraining corpora and improved tokenization, potentially reducing the impact of limited per-language training data and enhancing generalization to dialectal variants and code-mixing, (ii) lightweight calibration strategies (per-language thresholds and confidence-based abstention) to reduce unsafe dispatch under ambiguity and (iii) compact/distilled variants to further reduce latency and memory footprint for edge deployments.

Given ReChat's modular architecture, future work will focus on strengthening the NLP module under realistic usage conditions. In particular, we will evaluate stronger multilingual encoders and compact transformer variants within the same intent-to-action pipeline, and incorporate confidence-aware clarification strategies to avoid unsafe dispatch under ambiguous user commands. We also plan to extend the action layer with additional schedule-management operations (e.g., create/modify/delete schedules) and tariff-aware charging policies, while preserving deterministic execution and auditability. These extensions aim to improve robustness, broaden functional coverage, and further reduce interaction friction for multilingual EV users.

Author Contributions

Pablo Donate: conceptualization, data curation, formal analysis, investigation, methodology, software, visualization, writing – original draft. **Julio A. Sanguesa:** conceptualization, formal analysis, methodology, validation, writing – review and editing. **Piedad Garrido:** conceptualization, funding acquisition, project administration, supervision, writing – review and editing. **Vicente Torres – Sanz:** investigation, methodology, software, validation, writing – review and editing. **Francisco Martinez:** formal analysis, resources, validation, writing – review and editing. **Carlos Tavares Calafate:** funding acquisition, supervision, writing – review and editing.

Funding

This work has been partially supported by the Government of Aragón and the European Social Fund 'Construyendo Europa desde Aragón' (T40_23D Research Group), as well as by research project MORELLINO (CIPROM/2023/29), funded by 'Direcció General de Ciència i Investigació' Generalitat Valenciana - SPAIN.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

1. M. M. Islam, H. Shareef, and A. Mohamed, "Optimal Location and Sizing of Fast Charging Stations for Electric Vehicles by Incorporating Traffic and Power Networks," *IET Intelligent Transport Systems* 12, no. 8 (2018): 947–957, https://doi.org/10.1049/iet-its.2018.5136.

2. International Energy Agency, *Global EV Outlook 2024*, (International Energy Agency, 2024), https://www.iea.org/reports/global-ev-outlook-2024.

3. L. Hutchinson, B. Waterson, B. Anvari, and D. Naberezhnykh, "Potential of Wireless Power Transfer for Dynamic Charging of Electric Vehicles," *IET Intelligent Transport Systems* 13, no. 1 (2019): 3–12, https://doi.org/10.1049/ietits.2018.5221.

4. S. Syalini and M. Prathiba, "Electric Vehicle Charging Infrastructure: Status, Challenges and Future Directions," in *Empowering Business Through Technology: Innovations Shaping Our Future*, vol. 620 (Springer, 2026), 567–579, https://doi.org/10.1007/978-3-032-02056-7_46.

5. F. Elghitani and E. F. El-Saadany, "Efficient Assignment of Electric Vehicles to Charging Stations," *IEEE Transactions on Smart Grid* 12, no. 1 (2020): 761–773, https://doi.org/10.1109/TSG.2020.3016476.

6. A. O. Yontem, K. Li, D. Chu, V. Meijering, and L. Skrypchuk, "Prospective Immersive Human-Machine Interface for Future Vehicles: Multiple Zones Turn the Full Windscreen Into a Head-Up Display," *IEEE Vehicular Technology Magazine* 16, no. 1 (2020): 83–92, https://doi.org/10.1109/MVT.2020.3013832.

7. S. K. Dam, C. S. Hong, Y. Qiao, and C. Zhang, "A Complete Survey on LLM-Based AI Chatbots," preprint, arXiv, June 24, 2024, https://doi.org/10.48550/arXiv.2406.16937.

8. A. Abdellatif, K. Badran, and E. Shihab, "MSRBot: Using Bots to Answer Questions From Software Repositories," *Empirical Software Engineering* 25, no. 3 (2020): 1834–1863, https://doi.org/10.1007/s10664-019-09788-5.

9. R. K. Yadav, S. Harwani, S. K. Maurya, and S. Kumar, "Intelligent Chatbot Using GNMT, SEQ-2-SEQ Techniques," in *2021 International Conference on Intelligent Technologies (CONIT)* (IEEE, 2021), 1–5, https://doi.org/10.1109/conit51480.2021.9498485.

10. R. Ramakrishnan, P. Thangamuthu, A. Nguyen, and J. Gao, "Revolutionizing Campus Communication: NLP-Powered University Chatbots," *International Journal of Advanced Computer Science and Applications* 15, no. 6 (2024): 654–661, https://doi.org/10.14569/ijacsa.2024.0150606.

11. G. Zhang, J. Wu, G. Jeon, and P. Wang, "A Social Group Chatbot System by Multiple Topics Tracking and Atkinson-Shiffrin Memory Model Using AI Agents Collaboration," *Expert Systems* 42, no. 2 (2024): e13882, https://doi.org/10.1111/exsy.13766.

12. D. G. K. D. Gopisetty, "Chatbot Building With BERT for E-Commerce," *International Journal for Research in Applied Science and Engineering Technology* 12, no. 3 (2024): 2587–2592, https://doi.org/10.22214/ijraset.2024.59449.

13. D. Cook, D. Peters, L. Moradbakhti, et al., "A Text-Based Conversational Agent for Asthma Support: Mixed-Methods Feasibility Study," *Digital Health* 10 (2024): 20552076241233508, https://doi.org/10.1177/20552076241258276.

14. S. Kovacevic, T. Popovic, I. Jovovic, S. Cakic, and D. Babic, "Hotel Chatbot Receptionist for Smart Hospitality," in *2024 28th International Conference on Information Technology (IT)* (IEEE, 2024), 1–4, https://doi.org/10.1109/it61232.2024.10475740.

15. K. Kreimeyer, M. Foster, A. Pandey, et al., "Natural Language Processing Systems for Capturing and Standardizing Unstructured Clinical Information: A Systematic Review," *Journal of Biomedical Informatics* 73 (2017): 14–29, https://doi.org/10.1016/j.jbi.2017.07.012.

16. L. Laranjo, A. G. Dunn, H. L. Tong, et al., "Conversational Agents in Healthcare: A Systematic Review," *Journal of the American Medical Informatics Association* 25, no. 9 (2018): 1248–1258, https://doi.org/10.1093/jamia/ocy072.

17. M. P. Jacob, D. Dutt, D. Sankar N, O. Shinde, and S. Bhattacharya, "Chatbots For Customer Support," *International Journal of Emerging Technologies and Innovative Research* 8, no. 5 (May 2021): a165–a172.

18. E. Adamopoulou and L. Moussiades, "An Overview of Chatbot Technology," in *IFIP International Conference on Artificial Intelligence Applications and Innovations* (Springer, 2020), 373–383, https://doi.org/10.1007/978-3-030-49186-4_31.

19. E. Karpas, O. Abend, Y. Belinkov, et al., "MRKL Systems: A Modular, Neuro-Symbolic Architecture That Combines Large Language Models, External Knowledge Sources and Discrete Reasoning," preprint, arXiv, May 2, 2022, https://doi.org/10.48550/arXiv.2205.00445.

20. S. Yao, J. Zhao, D. Yu, et al., "ReAct: Synergizing Reasoning and Acting in Language Models," paper presented at the Eleventh International Conference on Learning Representations (2022), Kigali, Rwanda, May 1–5, 2023.

21. T. Schick, J. Dwivedi-Yu, R. Dessì, et al., "Toolformer: Language Models Can Teach Themselves to Use Tools," *Advances in Neural Information Processing Systems* 36 (2023): 68539–68551.

22. L. Zhang, Y. Ji, X. Li, et al., "Multi-Objective Charging Scheduling for Electric Vehicles at Charging Stations With Renewable Energy Generation," *Green Energy and Intelligent Transportation* 4, no. 4 (2025): 100283, https://doi.org/10.1016/j.geits.2025.100283.

23. D. Cui, Z. Wang, P. Liu, et al., "Coordinated Charging Scheme for Electric Vehicle Fast-Charging Station With Demand-Based Priority," *IEEE Transactions on Transportation Electrification* 10, no. 3 (2023): 6449–6459, https://doi.org/10.1109/TTE.2023.3334809.

24. N. K. Saxena and D. W. Gao, "Power Quality Enhancement by Mitigating Load Imbalance From Random Electric Vehicle Fleet at Electric Vehicle Charging Stations," *Green Energy and Intelligent Transportation* 4, no. 4 (2024): 100222, https://doi.org/10.1016/j.geits.2024.100222.

25. X. Li, Z. Wang, L. Zhang, et al., "A Comparative Study of Real-Time Coordinate Charging Schemes for Residential Electric Vehicles," *Journal of Energy Storage* 98 (2024): 113021, https://doi.org/10.1016/j.est.2024.113021.

26. M. Sundararaman and B. Sambasivam, "The Road to Net Zero in a Renewable Energy-Dominated Electricity System: Impact of EV Charging and Social Cost of Emission on the Optimal Economic Dispatch," *Green Energy and Intelligent Transportation* 4, no. 3 (2025): 100280, https://doi.org/10.1016/j.geits.2025.100280.

27. T. Fleck, S. Gohlke, and Z. Nochta, "A System for the Efficient Charging of EV Fleets," *World Electric Vehicle Journal* 14, no. 12 (2023): 335, https://doi.org/10.3390/wevj14120335.

28. A. Malkova, S. Striani, J. M. Zepter, M. Marinelli, and L. Calearo, "Laboratory Validation of Electric Vehicle Smart Charging Strategies," in *2023 58th International Universities Power Engineering Conference (UPEC)* (IEEE, 2023), 1–6, https://doi.org/10.1109/UPEC57427.2023.10294673.

29. P. Donate, J. A. Sanguesa, P. Garrido, and V. Torres-Sanz, "Enhancing Vehicular Charging Systems With a Natural Language HMI Solution," in *2025 34th International Conference on Computer Communications and Networks (ICCCN)* (IEEE, 2025), 1–6, https://doi.org/10.1109/ICCCN65249.2025.11133997.

30. R. Anitha and K. A. Kumar, "Sentiment Analysis in Low Resource Language: Exploring BERT, MBERT, XLM-R, and RNN architectures to Underpin the Deep Language Understanding," *Journal of Nonlinear Analysis and Optimization* 15, no. 2 (2024): 3, https://doi.org/10.1109/ACCESS.2024.3398635.

31. C. C. Tung, Z. H. Tan, Z. Y. Chook, C. W. Tan, and K. Y. Lim, "Leveraging NLP for Building Efficient Information Retrieval Systems: A Performance Analysis," in *2024 28th International Computer Science and Engineering Conference (ICSEC)* (IEEE, 2024), 1–6, https://doi.org/10.1109/ICSEC62781.2024.10770736.

32. M. M. Hossain, N. Nahid, G. Roy, M. A. N. Mojumder, and M. S. Rahman, "Optimizing Multilingual Summarization: Advanced Corpus Creation and Filtering Approach," in *2024 27th International Conference on Computer and Information Technology (ICCIT)* (IEEE, 2024), 2482–2487, https://doi.org/10.1109/ICCIT64611.2024.11022549.

33. K. Nimavat and T. Champaneria, "Chatbots: An Overview Types, Architecture, Tools and Future Possibilities," *International Journal of Scientific Research and Development* 5, no. 7 (2017): 1019–1024.