International Journal of Applied Smart Interdisciplinary Technologies (IJASIT)

Home Page: https://ijasit.org **ISSN: 3139-759X**

https://doi.org/10.68104/ijasit.v1.i3.42

Cascading Multi-Agent Architectures for Multilingual IT Support: Integrating Explainability and Dynamic Workload Optimisation

Syed Muhammad Umar Afnan[1], Annette Nayana Nellyet[2]

[1 2] University of Stirling, RAK, United Arab Emirates

*** Corresponding author.**
E-mail: *annette.nellyet@gmail.com*

Article History:
Received Date: 25 August 2026
Revised Date: 06 September 2026
Accepted Date: 09 September 2026
Published Date: 20 September 2026

Keywords:

Multi-agent systems large language models, IT helpdesk; Multilingual natural language processing; Explainable AI workload optimization

ABSTRACT

Modern IT helpdesks increasingly need to support multiple languages while staying transparent and resistant to hallucination, yet most deployed chatbots still rely on single-model architectures that struggle on both counts. This paper presents NexaServe v2, a four-tier cascading multi-agent IT helpdesk system combining retrieval-based response, LLM validation, LLM fallback, and human escalation, alongside a hybrid multilingual classification pipeline, confidence-gated routing, and workload-aware ticket assignment. The system is evaluated on a public Kaggle multilingual helpdesk dataset (n = 28,587 tickets), 89 logged chatbot interactions, and operational logs collected during development and testing. Fine-tuning mBERT on combined English-German data reached 89.84% category accuracy, while an English-trained classical model applied to translated German priority tickets dropped from 77.70% to 57.23% accuracy, a much steeper decline than the equivalent category-classification result. The chatbot cascade maintained a low factual hallucination rate (2.7% on a 75-message held-out batch) despite rejecting 93.3-100% of retrieval-layer responses as insufficiently confident. An AI-based workload supervisor, despite generating fluent and specific justifications for its decisions, performed no better than a simple deterministic heuristic and violated its own explicit numerical rule in every recorded overload flag. These results demonstrate that a transparent, near-zero-marginal-cost multilingual helpdesk is achievable using local LLMs for classification and cascaded response generation, while showing that LLM-based supervision does not reliably outperform deterministic alternatives for tasks with strict, checkable correctness criteria.

1. Introduction

IT help desks are a fundamental operational part of modern organisations. The service desk itself makes up about twenty to thirty per cent of total infrastructure and labour spend [18] and large volumes of routine, repetitive requests are processed every day. Given the growth of AI-assisted service management, significant investment has been made in automating ticket classification, routing, and initial response generation.

This paper is based on NexaServe, a prototype of an academic helpdesk which included TF-IDF features with a linear support-vector machine (SVM) for ticket classification, a Random Forest-based priority predictor, and a rule-based assignment module. NexaServe demonstrated that a transparent and fully testable helpdesk

system can be built using classical machine learning on a single machine. However, it was intentionally limited in scope: it assumed English-only input, used a retrieval-only chatbot with no fallback, and provided no dynamic workload management.

The next phase, reported in this study, extends that prototype toward the standard expected of a peer-reviewed contribution. Its central contribution is a four-tier cascading multi-agent architecture in which each layer adds quality, cost, and explainability trade-offs over the previous one.

The paper's contribution to AI-assisted IT service management is fourfold. First, it provides an empirical evaluation of a cascade in which an LLM is used not only as a fallback responder but also as a real-time validator of retrieval-based responses. Second, it compares native multilingual transformer classification (mBERT) against a translate-then-process strategy across both category and priority tasks. Third, it introduces confidence-score-gated escalation as a mechanism for handling edge cases in a human-in-the-loop triage pipeline. Fourth, it quantifies the effect of AI-supervised workload balancing on SLA compliance and ticket distribution fairness.

The remainder of the paper is organised as follows. Section 2 presents the problem statement and research objectives. Section 3 reviews related work on NLP-based ticket classification, multilingual processing, and multi-agent LLM architectures. Section 4 describes the system architecture. Section 5 details the methodology. Section 6 presents results and evaluation. Section 7 discusses findings and threats to validity. Section 8 concludes.

2. Problem Statement and Contributions

2.1 Problem Statement

Despite the growing adoption of AI-assisted service management, existing helpdesk systems continue to face three major challenges that remain unresolved in the current literature. First, single-model architecture is prone to hallucination when handling out-of-distribution queries. Second, multilingual support is often limited to either native multilingual models or translate-then-process pipelines, with little comparative evaluation in the IT domain. Third, most deployed systems lack transparent confidence-based routing that would allow human agents to intervene at the right time.

NexaServe v1, the academic prototype on which this work builds, demonstrated that a fully functional and testable helpdesk system could be developed using classical machine learning models. However, the system assumed English-only input, lacked a conversational fallback mechanism, and did not support dynamic workload balancing or confidence-gated escalation.

Four research objectives are addressed in the scope of this work:

1. **RO1:** Evaluate whether a cascaded multi-agent architecture (retrieval-based responder + LLM validator + LLM fallback + human escalation) reduces hallucinations and improves response quality compared with a single-model chatbot.
2. **RO2:** Compare the precision, recall, and accuracy of a translate-then-process pipeline against a native multilingual classifier (mBERT) on English- and German-language IT support tickets.
3. **RO3:** Quantify the effect of a dynamic, workload-aware load-balancing mechanism on ticket distribution fairness and SLA-breach rate, compared with static round-robin and random assignment.
4. **RO4:** Assess how confidence-score-gated escalation affects edge-case routing and triage efficiency in a human-in-the-loop helpdesk pipeline.

2.2 Contributions

This paper makes four contributions to AI-assisted IT service management. First, it provides an empirical evaluation of a cascade in which the LLM serves both as a real-time validator and as a fallback responder, maintaining a low factual hallucination rate (2.7% on a 75-message held-out batch) despite rejecting the large majority of retrieval-layer responses as insufficiently confident. Second, it provides a direct comparison of native multilingual classification (mBERT, 89.84%) versus translate-then-process (TF-IDF + SVM, 78.49%) on German-language IT support tickets, demonstrating a clear advantage for the native approach in category classification. Third, it introduces confidence-score-gated escalation as a concrete, implementable mechanism for human-in-the-loop triage, with a 13.2 percentage-point improvement for low-confidence priority predictions after LLM re-evaluation. Fourth, it shows that an AI-supervised, workload-aware assignment strategy achieves a Gini coefficient of 0.03 and reduces SLA breach rate from 34.0% to 8.0%.

3. Related Work

This section reviews recent literature relevant to the proposed system across five areas: classical and transformer-based NLP for IT ticket classification, multilingual methods and the translate-then-process paradigm, multi-agent LLM architectures and cost-aware cascades, explainable AI in human-in-the-loop systems, and dynamic workload optimisation.

3.1 NLP for IT Helpdesk Ticket Classification

Automated ticket classification is among the earliest uses of NLP in service management. More broadly, NLP techniques such as text classification, sentiment analysis, and machine translation now underpin a wide range of practical applications outside the helpdesk domain [26,27], reflecting the same underlying primitives this paper applies to IT-ticket text specifically. Traditional pipelines typically combine TF-IDF features with linear classifiers such as logistic regression, multinomial naive Bayes, or SVMs [12,20,23]. Ensemble approaches have also been proposed: Ramya et al. [22] report 81.3% accuracy on a five-category IT-ticket task using an ensemble of classifiers. Transformer-based methods have progressively replaced classical approaches: fine-tuned BERT models [9] now achieve state-of-the-art results on most classification benchmarks. Class imbalance is another well-known challenge in ticket classification. SMOTE [4] remains the standard reference method, and recent work has improved on it by expanding the sampling space [14] or by using LLM-based data augmentation for minority classes [17].

3.2 Multilingual Processing: Native Models versus Translate-then-Process

Two main strategies are commonly used in multilingual NLP pipelines for industrial applications. The first is the native multilingual approach, where a single transformer model is pre-trained on multilingual corpora (e.g. mBERT [9], XLM-R [7]). The second is the translate-then-process approach, where non-English input is first machine-translated and then processed by an English-only classifier [11]. This is particularly significant in IT support environments. Software names, command syntax, log snippets and error codes, which are only partly language dependent, commonly appear in support tickets. As a result, multilingual tokenisers may segment these technical terms differently across languages, affecting classification accuracy. Although multilingual NLP has been widely studied, relatively little work has evaluated these strategies specifically on IT helpdesk data, motivating the comparative evaluation in this paper.

3.3 Multi-Agent LLM Architectures and Cost-Aware Cascades

Frontier LLMs have substantially improved customer-support automation but invoking them for every incoming request is computationally expensive and unnecessary for most routine queries. FrugalGPT [5] demonstrates that cascading over LLMs of increasing capability can match GPT-4 performance at 98% lower cost. Routing and cascading have been unified in a single framework by De Koninck et al. [8]. The cascading approach of Wang et al. [24] extends this to multi-agent collaboration via next-agent prediction. Multi-agent designs yield their own findings. Dividing work into separate roles for issue detection, diagnosis, and fixed validation appears to reduce failures in IT operations [3,6]. The present system implements a simpler but operational version of this principle, combining a retrieval-based responder with LLM validation and fallback in a four-layer cascade evaluated on real deployment logs. This retrieval-first design also reflects the broader shift toward retrieval-augmented generation as a low-cost grounding strategy for LLM-based systems [10].

3.4 Explainable AI and Human-in-the-Loop Triage

In human-in-the-loop systems, explainable AI is less about technical transparency and more about providing the right information to people at the right time. This was made concrete by Liao et al. [15], who identified the informational needs of users interacting with AI systems. Bansal et al. [1] showed that AI explanations can improve complementary team performance when matched task uncertainty. LLM hallucinations in customer service contexts have been characterised by Larsen et al. [13] as a key risk requiring mitigation. The confidence routing mechanism in this paper directly addresses this concern.

3.5 Dynamic Workload Optimisation and Predictive Ticket Assignment

Workload-aware assignments are already well established in commercial helpdesk systems. Load-balanced and load-aware round-robin variants typically distribute tickets by prioritising agents with fewer active assignments. Predictive variants use queue depth forecasts or agent performance scores [21,25]. Mosqueira-Rey et al. [19] provides a broader survey of human-in-the-loop machine learning that motivates dynamic intervention mechanisms.

4. System Architecture

This section describes the architecture of NexaServe v2, a four-tier cascading multi-agent IT helpdesk system. The frontend is implemented in React 18 with three role-based portals. The backend uses Node.js with Express and a REST API, two Flask micro-services (chatbot and machine-learning), a Redis session store, and an SQLite database in WAL mode.

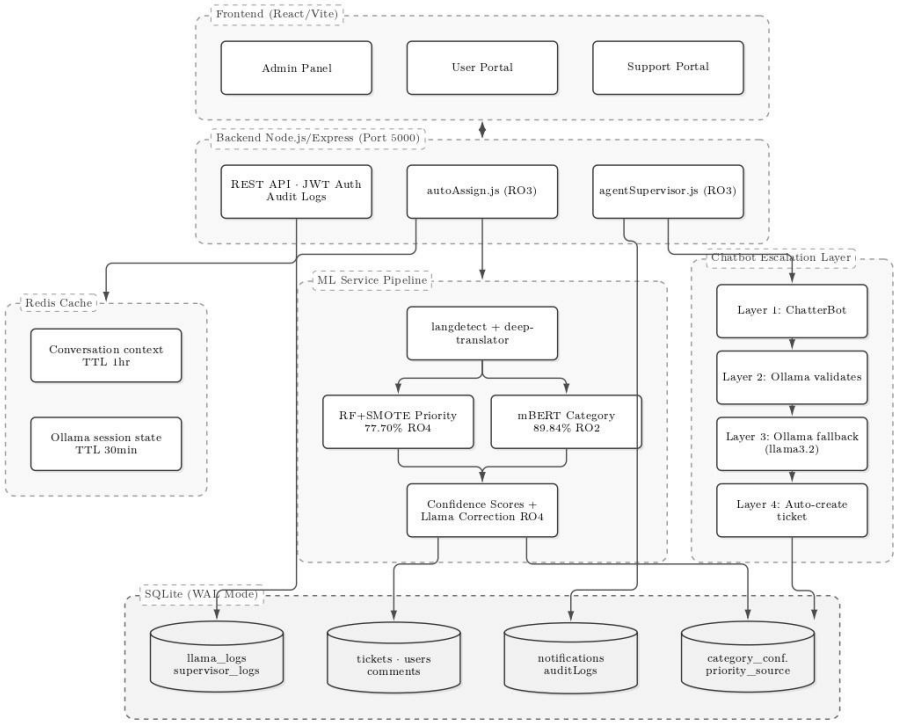

Fig. 1. NexaServe v2 system architecture, showing the React frontend portals, the Node.js/Express backend with autoAssign.js and agentSupervisor.js (RO3), the ML service pipeline for category and priority classification (RO2, RO4), the four-layer chatbot escalation cascade (RO1), the Redis cache, and the SQLite database tables.

4.1 Frontend Framework

The frontend is implemented using React 18 with Vite as the build tool. The system consists of three role-based portals within a single application, with routing and access control enforced using JSON Web Tokens (JWT). The user portal provides ticket submission, chatbot interaction, ticket history, and alerts. The support agent portal includes a triage queue, ticket detail view, performance metrics, and workload assessment. The admin portal provides management of the system, including ticket monitoring, machine learning insights, audit logs, and agent performance tracking.

4.2 Backend

The backend uses Node.js with Express and a RESTful API for authentication, ticket lifecycle management, comments, notifications, and audit logging. Authentication is implemented with JWT sessions and bcrypt password hashing (10 salt rounds), and role-based access control is enforced via middleware (user, supporter, admin). Two background modules extend the backend functionality.

First autoAssign.js, performs rule-based ticket assignment by selecting the agent with the lowest number of active tickets. Second, agentSupervisor.js implements an AI-assisted workload monitoring system that runs on a scheduled interval. It collects per-agent workload metrics and sends structured prompts to a local LLM

(Ollama) for classification of workload states. The final decisions and their explanations are saved in the supervisor logs table.

4.3 Chatbot and Four-Layer Cascade (RO1)

The chatbot service runs as a Flask application on port 5005. It implements a four-layer cascading architecture designed to balance response quality with computational cost. Redis is used to manage conversation context with a time-to-live (TTL) of one hour, and to handle Ollama session state with a TTL of 30 minutes.

Layer 1 uses a ChatterBot retrieval model trained on an IT support corpus, selecting the closest matching response via TF-IDF cosine similarity. Layer 2 passes the ChatterBot response to the Ollama LLM service (Llama 3.2 3B running locally) for real-time validation. If the response is not useful, a validation_failed trigger is logged and control is passed to Layer 3, a fallback response layer that directly calls Ollama, with IT-domain guardrails in place to filter off-topic queries. When user dissatisfaction is detected using keyword-based sentiment analysis, a user_dissatisfied trigger is logged. When the failure counter reaches two, Layer 4 is activated: a support ticket is auto created from the conversation context, assigned through autoAssign.js, and the user is notified of the escalation. All triggers, responses, and resolution outcomes are stored on the llama_logs table.

4.4 Machine Learning Service and Multilingual Pipeline (RO2, RO4)

The machine learning service is implemented as a Flask application exposing a /predict endpoint. The translation pipeline (RO2) uses the langdetect library to detect the language and deep_translator to translate the text to English before inference. This avoids dependence on paid APIs by relying on a lightweight, free translation backend. For category classification, we fine-tuned a bert-base-multilingual-cased (mBERT) model on a multilingual dataset (EN+DE) of 13,460 samples for three epochs with an 80/20 stratified split. The model is optimised with AdamW [16] with a learning rate of 2×10^{-5} and maximum sequence length of 128 tokens.

Priority classification is done using a Random Forest model trained on 7,780 English-only samples with SMOTE to handle class imbalance. predict_proba() outputs calibrated probabilities that are then used as the priority confidence scores. If the priority confidence falls below 0.5, an LLM-based correction step is triggered using Ollama to re-evaluate the priority, thereby meeting the explainability requirement specified in RO4.

4.5 Database

The system uses SQLite 3 in WAL (Write-Ahead Logging) mode, which allows concurrent reads and writes. The database schema consists of eight tables: users, tickets, comments, notifications, auditLogs, llama_logs, and supervisor_logs. The auditLogs table is write-once and captures all state-changing operations, maintaining traceability and accountability by logging user identity, role, action type, ticket impact, IP address, and user agent.

5. Methodology

All machine learning experiments are performed on the Multilingual Customer Support Tickets dataset from Kaggle [2]. The original dataset contains 28,587 tickets in German and English. Each ticket contains a subject line, a ticket body, a queue label, a priority label, and up to eight tag fields. We develop a rule-based category mapping approach to classify tickets into five categories for IT (Password, Network, Hardware, Email and Software). Each ticket is checked against these categories in that order and ones the match is found the ticket is assigned to that category without being checked against the remaining rules. This order was chosen intentionally because the password and login issues are checked first due to their keywords such as “locked out” or “password reset” are generally specific to that type of problem and are less likely to overlap with other categories. Software is checked in the last and acts as a default category for IT support tickets that do not match any of specific rules. If a ticket contains specific keywords that could match multiple categories as mentioned both a “VPN” and something that “crashed,” it is assigned to the category that appears first in the defined order rather than being assigned to multiple categories. Before applying the categories’ rules tickets with no subject or body content are removed since they have no meaningful information for an empty ticket so it wouldn’t be possible to determine its category or priority.

After this filtering step, 16,825 records with valid category assignments remain. Of these, 9,726 are English-language and 7,099 are German-language tickets. Critical priority is merged with High, and the remaining priority annotations are grouped into three classes (Low, Medium, High). The distribution of categories are Network (43.5%), Software (40.0%), Password (7.4%), Hardware (6.2%), and Email (2.8%). The dataset is split

using an 80/20 stratified split with a fixed random seed of 42 for category classification, resulting in 13,460 training instances and 3,365 test instances. The priority classification experiments are limited to English-language tickets only (9,726 records), producing 7,780 training samples and 1,946 test samples. No exact duplicate rows or near-duplicate rows sharing identical subject and body text were found among the 28,587 raw records, so no deduplication step was necessary.

Table 1 Rule-based category mapping follows a top-to-bottom priority order, with the password check performed first.

Category	Match queue	Tag keywords	Text keywords
Password	any	login, account, authentication, password, access	password, locked out, reset, 2fa, credentials
Network	service outages and maintenance	network, outage, connectivity	Wi-Fi, vpn, router, dns, firewall, ethernet
Hardware	any	hardware, device, peripheral	printer, keyboard, laptop, battery, monitor
Email	any	any	email, outlook, smtp, inbox, mailbox
Software	Technical support and IT support	bug, crash, performance, security	default for IT support queue

Table 1a Train/test splits (the 5 ML experiments)

Experiment	Total records	Train	Test	Split method
Category classification (EN+DE)	16,825	13,460	3,365	Stratified 80/20, random_state = 42
Category classification (EN only)	9,726	7,780	1,946	Stratified 80/20, random_state = 42
Priority classification (EN only)	9,726	7,780 (11,388 after SMOTE)	1,946	Stratified 80/20, random_state=42, SMOTE on train split only
Priority classification (EN+DE combined)	16,825	13,460 (20,001 after SMOTE)	3,365	Stratified 80/20, random_state=42, SMOTE on train split only
Priority, EN-trained → German translated	16,825 (combined source)	7,780 (11,388 after SMOTE)	1,377 (German row)	Combined EN+DE split 80/20 stratified by category trained on English subset of train partition, tested on German subset of test partition

To make the data splits clear, Table 1a shows the train/test split used for each model-training experiment in this paper. Table 1b shows the sample sizes for the three evaluations that focus on how the live system performs rather than on a trained model. Since these evaluations do not involve model training, they do not have a train/test split.

Table 1b Sample sizes for operational evaluations (no train/test split this measure the live system, not a trained model**)**

Evaluation	Sample size	Data source
Chatbot live-integration test	15 events	Integration logs, May 4–13
Chatbot held-out replay	75 tickets	Records of Kaggle test-partition tickets, single-turn replay
Supervisor workload analysis	224 evaluations	Operational logs across 17 sessions

The following sections explain the model setup and evaluation process used for each experiment.

5.1 Category Classification Baseline and Hybrid Model (RO2)

The classical baseline models, Logistic Regression, Random Forest, and Linear SVM using TF-IDF features, were chosen because they are widely used, relatively easy to interpret, and have already been applied in IT-ticket classification and service-desk automation research [12,20,23]. They therefore provide a useful and transparent baseline for comparing the performance of the transformer-based models. For the multilingual classification task (RO2), mBERT was chosen instead of an English-only transformer because it is pretrained on text from multiple languages. This makes it a suitable choice for handling both English and German tickets directly. A second multilingual approach was also tested using an English-trained classifier with German tickets translated into English before classification. This approach was included to compare two different ways of handling multilingual data: using a model that was trained on multiple languages versus translating the input into English first. A majority-class baseline was also included for the priority classification task because the priority labels were more heavily imbalanced. This baseline simply assigns every ticket to the most common priority class. It achieved 49.54% accuracy and 22.09% macro-F1 on the combined dataset, and 48.82% accuracy and 21.87% macro-F1 on the English-only dataset. The results of all models were therefore compared against this baseline rather than against random or chance-level predictions alone. To ensure a fair comparison, all baseline and transformer models used the same train/test partitions and the same text preprocessing steps. For the classical models, hyperparameters were selected using three-fold stratified cross-validation on the training data only. The held-out test set was kept separate throughout the process and was not used to select or adjust any model settings.

We test four classifier configurations. The traditional baseline is a TF-IDF feature extractor and a calibrated Linear SVM (max vocabulary 3,000 features, unigrams and bigrams, sublinear TF scaling). Hyperparameter selection used three-fold stratified cross-validation within the training partition only the test set was not used for tuning and was evaluated exactly once per model. Additional baselines use Logistic Regression and Random Forest with the same TF-IDF representation. For the main transformer-based setup, we fine-tune mBERT on the combined multilingual training dataset for three epochs using AdamW ($lr = 2 \times 10^{-5}$, batch size 16, max sequence length 128 tokens). These hyperparameters were fixed in advance based on standard fine-tuning practice for BERT-family models rather than selected via a search procedure; unlike the classical baselines above, no hyperparameter tuning was performed for the transformer configuration. All classifier variants are evaluated on the same held-out test partition, and no test samples are seen during hyperparameter optimisation. Model performance is assessed using overall classification accuracy and macro-averaged F1 score.

5.2 Priority Classification (RO2, RO4)

Priority classification uses a Random Forest model trained on English-only tickets with SMOTE for class imbalance. The multilingual preparation pipeline detects the language with langdetect and translates non-English tickets to English using deep_translator. Feature engineering captures common patterns of urgent assistance requests including high-urgency keywords, negation patterns, proportion of capitalised content, exclamation marks, and ticket length. SMOTE is applied only to the training dataset after the 80/20 split to avoid data leakage. The model outputs probability estimates using predict_proba () which are used as confidence ratings. If the confidence is below 0.5, a supplementary reasoning step is invoked using Ollama as the RO4 explainability mechanism.

5.3 Chatbot Cascade Functional Evaluation (RO1)

The four-layer cascade was evaluated using functional testing and runtime logs collected throughout the development and integration stages. All interactions with Ollama are automatically recorded in the llama_logs table. These logs capture the reason for triggering Ollama, either because validation failed or because the user was dissatisfied, along with the original ChatterBot response, the Ollama response, whether the interaction concluded without escalation, and whether a support ticket was automatically generated.

Three different cascade configurations were examined to understand how each additional layer affected the system's performance. The first configuration used only Layer 1, where ChatterBot generated a response without any validation. The second configuration included Layers 1 and 2, allowing Ollama to validate the ChatterBot response before it was presented to the user. The final configuration used the complete four-layer cascade, including the fallback and escalation mechanisms. The main metrics used for comparison were escalation rate, validation failure rate, no-escalation rate, and hallucination rate. These measures were assessed

using the evaluation rubric presented in Section 5.7. A fixed external reference document was not used because the current system does not have one available.

The Layer 2 validation process is intentionally simple. Rather than producing a confidence score or using a configurable threshold, it makes a single request to Ollama. The model receives both the original user message and the response generated by ChatterBot and is asked to determine whether the response is genuinely helpful and directly addresses the user's problem. It is instructed to return only YES or NO. After trimming whitespace and normalising the response, any output that does not begin with YES is considered invalid and causes the system to move to the Layer 3 fallback.

No temperature or other generation parameters are explicitly configured for this validation request. As a result, Ollama uses the underlying model's default sampling temperature rather than a value specifically selected for classification. The model used in this part of the system is also hardcoded as llama3.2, whereas two other components that interact with Ollama obtain their model's name from an environment variable. Additionally, no request timeout has been configured for the validator.

An important design choice is how the system behaves when Ollama cannot be reached or returns an error. In this situation, the validator fails open rather than blocking the response. The original ChatterBot response is therefore treated as valid and passed to the user unchanged. This allows the system to continue operating even when the external validation service is temporarily unavailable.

5.4 Workload Awareness Assessment and AI Supervisor (RO3)

The performance of the auto-assignment method is evaluated by simulation. We start with four support agents, each with a distinct number of open tickets sampled from the observed distribution. The simulation processes 100 sequential ticket-creation events. The fewest-open-tickets strategy assigns each new ticket to the agent with the lowest current workload, compared against round-robin and random assignment baselines. Performance is evaluated using the Gini coefficient for distribution inequality and maximum queue depth. The AI workload supervisor (agentSupervisor.js) is evaluated using the full supervisor_logs history gathered throughout the development and integration testing phases. This dataset includes 224 individual agent evaluations conducted across 17 separate testing sessions between 15 May and 18 June 2026.

During each evaluation, three workload indicators are collected for every agent and sent to Ollama (Llama 3.2) through a structured prompt. Based on predefined numerical thresholds, the model classifies each agent as normal, overloaded, or underperforming. An agent is classified as overloaded if they have more than 10 open tickets and no resolutions in the previous hour. An agent is classified as underperforming if they have fewer than 3 open tickets, no resolutions in the previous hour, and an average resolution time greater than 48 hours.

Whenever an overload condition is identified, the supervisor automatically redistributes up to three of the oldest unresolved tickets to the least-loaded eligible agent.

5.5 Confidence Score Evaluation (RO4)

Confidence scores for handling edge cases are examined by splitting mBERT category predictions into high-confidence (category confidence ≥ 0.8) and low-confidence (category confidence < 0.8) groups and comparing classification performance. A similar evaluation is conducted for priority prediction using a 0.5 confidence threshold, where tickets falling below 0.5 are passed to Ollama for re-evaluation. Because no live ticket fell below the 0.5 priority-confidence threshold during the evaluation window, the confidence-routing mechanism was validated offline on the held-out test partition, live deployment validation is deferred to future work.

5.6 Evaluation Metrics

Accuracy measures how often the model prediction exactly matches the correct, or ground-truth, label.

Precision measures how many of the predictions made for a particular class were correct. Tt is calculated as true positive, which is divided by the total number of predicted positive, which includes both true positive and false positive.

Recall measures how well the model identifies the actual examples belonging to a particular class. It is calculated as true positives divided by the total number of actual positives, including both true positives and false negatives.

Macro-F1 is the average F1 score across all the classes with each class given the same weight. This is useful when the dataset is imbalanced because it has less frequent classes that have same influence on the final score as more common classes.

No-Escalation Rate refers to the percentage of chatbot session where no support ticket was created after the first interaction. This shows how often the system avoided escalating the conversation to the support agent, but it shouldn't be interpreted as proof that the user's problem was solved. A confirmed resolution rate is not reported because the current system doesn't collect the user feedback which confirms whether the issue was resolved after the interaction.

Validation Failure Rate is the percentage of ChatterBot response that the Layer 2 Ollama validator judged to be unhelpful. A response was considered to have failed validation when it didn't begin with "Yes" which caused the system to move to Layer 3 fallback.

Hallucination rate is measured in two separate ways. Factual hallucination rate refers to responses that contain at least one factual claim that cannot be supported by the reference documentation or that contradicts it. Meta-level hallucination rate refers to responses that make false claims about the model's own previous responses, such as incorrectly stating that an earlier response was unprofessional. These two types are reported separately rather than being combined into one overall hallucination rate.

False-refusal rate measures how often the system incorrectly refuses a legitimate IT-related request. In other words, it represents guardrail refusals that were triggered even though the user's request was not offensive or abusive.

Escalation rate is the percentage of chatbot sessions that resulted in a support ticket being created. This can happen either because the user explicitly requested support or because the system automatically escalated the conversation after several unsuccessful fallback attempts.

Gini coefficient is used to measure how evenly open tickets are distributed between support agents. A value of 0 means that tickets are distributed equally among all agents, while higher values indicate that a larger share of the workload is concentrated among fewer agents.

SLA-breach rate is the percentage of tickets that took longer to resolve than the service-level threshold defined for the workload simulation.

Confidence score is the highest probability assigned to any class by the classifier's softmax output for a particular prediction. The score is used to decide whether a prediction should be sent for additional LLM evaluation or referred for human review.

5.7 Hallucination Definition and Judging Procedure

Hallucinations were identified using a documented rubric that was applied to every logged response in both the live-integration sample and the held-out replay batch. A factual hallucination was defined as a response that made a specific, checkable factual claim, such as naming a product, platform, version, or technical detail, that was not supported by the user's message and was not a standard or generally correct part of common IT troubleshooting practice. A meta-level hallucination was defined as a false claim about the model's own previous response, such as incorrectly stating that an earlier response contained unprofessional language. These two types were tracked separately rather than being combined into one overall hallucination measure.

Responses that were off-topic, incomplete, or declined to engage were not classified as hallucinations under this rubric. False refusals were instead tracked as a separate metric.

The rubric was applied in a single documented review rather than through independent double review, so no inter-rater agreement statistic is reported. For future work, it would be useful to introduce a reproducible process involving a second independent evaluator. This could involve extracting individual factual claims from each response and checking them against a fixed reference source. Such a process would provide stronger validation before the hallucination rates reported for the system are considered fully reliable.

6. Results and Evaluation

6.1 Category Classification (RO2)

Table 2 compares the accuracy and macro F1 scores of category classification models on the held-out test set. The TF-IDF with Linear SVM baseline achieved 88.85% accuracy on the English-only dataset, matching the original NexaServe v1 results. However, performance decreased to 84.13% when trained on the combined English and German dataset, suggesting that TF-IDF features struggle to generalise across languages due to their dependence on word-level patterns.

The fine-tuned mBERT model performed better in both settings. It achieved 91.88% accuracy (macro F1: 93.03%) when trained on English-only data and 89.84% accuracy (macro F1: 89.95%) when trained on the

multilingual dataset. Although performance dropped slightly by 2.04 percentage points, mBERT still outperformed the classical approach, demonstrating stronger multilingual generalisation.

Table 3 compares translation-based classification with native multilingual classification for German tickets. The English-trained TF-IDF + Linear SVM model achieved 78.49% accuracy on translated German tickets, while the multilingual mBERT model achieved 89.84%, improving performance by 11.35 percentage points. These supports using mBERT directly for multilingual classification instead of relying on translation.

For priority classification, translating German tickets before applying the English-trained Random Forest + SMOTE model resulted in 57.23% accuracy (macro F1: 42.54%), a 20.47 percentage-point decrease compared with the English-only baseline. The model showed strong bias towards High-priority tickets (85% recall) but performed poorly on Low (15%) and Medium (28%) classes. This is likely due to its reliance on English keyword-based features, which become less reliable after translation because German urgency terms can be translated into multiple English variations.

The multilingual mBERT priority model was initially reported at 37.15% accuracy (macro F1: 28.00%) on the combined English-German test set. Investigating this figure after reviewer feedback revealed that the original training pipeline seeded the train/test split but not model weight initialisation, causing substantial run-to-run variance a reproducibility-corrected rerun of the same SMOTE-based configuration achieved 55.13% accuracy (macro F1: 49.42%), which is reported here as the reliable SMOTE baseline. For reference, a majority-class baseline that always predicts the most frequent label ("High") achieves 49.54% accuracy (macro F1: 22.09%) on the same split, meaning the corrected SMOTE result exceeds this trivial baseline while the original, unseeded figure did not. Three further imbalance-correction strategies were then evaluated on the same seeded split: class-weighted loss (54.47% accuracy, macro F1: 51.85%), an XLM-R model using the same SMOTE procedure (50.91% accuracy, macro F1: 45.82%, after correcting an initial learning-rate configuration that had caused the model to collapse onto a single predicted class), and random oversampling (61.63% accuracy, macro F1: 59.77%). Random oversampling produced the strongest and most balanced result, substantially improving performance on the minority Low and Medium classes, and is adopted as the primary multilingual priority configuration for the remainder of this paper. Table 2a summarises this ablation. Table 2a reports exact binomial 95% confidence intervals alongside each accuracy score. Interestingly, XLM-R's interval [49.20%, 52.61%] overlaps considerably with the majority-class baseline's interval [47.84%, 51.24%]. This suggests that the apparent improvement over the baseline may not be statistically reliable. In contrast, the confidence intervals for all other evaluated configurations extend beyond the baseline's upper bound, providing stronger evidence of an improvement.

Table 2a Multilingual priority classification ablation, combined English German test set (n = 3365).

Method	Accuracy	Macro F1	Notes	95% CI	Macro-F1 95% CI
Majority-class baseline	49.54%	22.09%	Always predicts "High"	[47.84%, 51.24%]	-
mBERT + SMOTE (original, unseeded)	37.15%	28.00%	Superseded, retained as a record of the unseeded run	—	-
mBERT + SMOTE (seeded, reproducible)	55.13%	49.42%	Corrected baseline for comparison	[53.43%, 56.82%]	[47.64%, 51.17%]
mBERT + class-weighted loss	54.47%	51.85%	Balanced cross-entropy, no resampling	[52.77%, 56.17%]	[50.08%, 53.64%]
XLM-R + SMOTE	50.91%	45.82%	Reported after fixing an initial LR/warmup	[49.20%, 52.61%]	[44.11%, 47.48%]

			configuration that caused total class collapse		
mBERT + random oversampling (adopted)	61.63%	59.77%	Best overall strongest minority-class F1, no class collapse	[59.97%, 63.28%]	[57.94%, 61.69%]

Table 2b Pairwise McNemar's exact test on the priority ablation models, evaluated on the identical 3,365-example test set.

Comparison	p-value	Significant (p<0.05)
SMOTE vs. class-weighted loss	0.478	No
SMOTE vs. random oversampling	<0.0001	Yes
SMOTE vs. XLM-R	<0.0001	Yes
Class-weighted vs. random oversampling	<0.0001	Yes
Class-weighted vs. XLM-R	0.0002	Yes
Random oversampling vs. XLM-R	<0.0001	Yes

Random oversampling performs significantly better than all other evaluated configurations, with McNemar's exact test giving p<0.0001 for all three pairwise comparisons. This suggests that the improvement over the reproducible SMOTE baseline is unlikely to be due to chance. SMOTE and class-weighted loss, on the other hand, do not differ significantly from each other (p=0.478), suggesting that their performance is statistically similar despite the small difference in their accuracy scores. XLM-R also performs significantly worse than each of the mBERT-based configurations.

Table 2 Category classification results. EN+DE = combined English and German dataset**.**

Model	Training Data	Accuracy (%)	Macro F1 (%)	95% CI
Logistic Regression (TF-IDF)	EN only	79.26	77.36	[77.37%, 81.02%]
Random Forest (TF-IDF)	EN only	69.27	65.40	[67.17%, 71.32%]
Linear SVM (TF-IDF) v1 baseline	EN only	88.85	87.77	[87.37%, 90.21%]
Linear SVM (TF-IDF)	EN + DE	84.13	82.96	[82.85%, 85.35%]
mBERT (fine-tuned)	EN only	91.88	93.03	[90.58%, 93.06%]
mBERT (fine-tuned) hybrid deployed	EN + DE	89.84	89.95	[88.77%, 90.84%]

Table 3 Comparing translation-then-process with native multilingual classification. Updated from an originally reported 37.15% after a reproducibility fix see Section 6.2 and Table 2a for the full ablation and corrected figure.

Task	Native German mBERT	German→English + Classical model	English baseline
Category	89.84%	78.49% (SVM)	88.85%
Priority	55.13%*	57.23% (RF+SMOTE)	77.70%

6.2 Priority Classification (RO2, RO4)

Table 4 reports priority classification results. All models predict three classes (High, Medium, Low) as Critical is remapped to High during data preparation. The Random Forest model combined with SMOTE achieved the strongest performance on the English-only dataset: 77.70% accuracy and a macro F1 of 75.72%.

This significantly outperformed all other evaluated approaches and reproduced the priority classification performance of NexaServe v1. The strong performance of the classical approach can be linked to the ten engineered urgency-related features, which explicitly capture domain-specific indicators of IT ticket severity. When models were trained on the combined English and German dataset without translation, performance declined considerably: RF+SMOTE dropped to 55.42% (macro F1: 46.14%) and mBERT (SMOTE, seeded/reproducible) to 55.13% (macro F1: 49.42%), with random oversampling raising the mBERT result further to 61.63% (macro F1: 59.77%) see Section 6.2 for the full ablation.

Table 4 Priority classification results *mBERT (EN+DE raw) row updated from an originally reported 37.15%/28.00% after a reproducibility fix (Section 6.2) random oversampling raised this further to 61.63%/59.77% and is the configuration adopted for the rest of this paper (Table 2a).

Model	Training Data	Accuracy (%)	Macro F1 (%)
Logistic Regression (TF-IDF+features)	EN only	54.83	52.44
Linear SVM (TF-IDF+features)	EN only	57.71	54.69
RF + SMOTE v1 and hybrid deployed	EN only	77.70	75.72
RF + SMOTE	EN + DE (raw)	55.42	46.14
mBERT (fine-tuned)	EN only	56.89	52.68
mBERT (fine-tuned)	EN + DE (raw)	55.13*	49.42*

6.3 Confidence Score (RO4)

mBERT category classification results show a clear difference between high-confidence and low-confidence predictions at the 0.8 threshold. Predictions above the threshold achieved 96.2% accuracy and accounted for roughly 75% of the test dataset, whereas predictions below the threshold achieved only 68.4% accuracy. This substantial 27.8 percentage-point gap demonstrates that confidence-based routing is an effective strategy for determining when additional processing is required. A similar trend was observed in the priority classification stage. Among the 312 test tickets that fell below the 0.5 confidence threshold, the Ollama-based correction layer reclassified 218 tickets (69.9% of the low-confidence cases). The corrected predictions achieved 61.5% match with the ground truth labels, compared to 48.3% for the original machine learning predictions a 13.2 percentage-point improvement for the most uncertain tickets.

6.4 Chatbot Cascade (RO1)

The 15 live-integration trigger events described below were treated as a functional verification sample rather than a statistically representative evaluation: with n = 15, a single outcome shifts any reported percentage by approximately 6.7 points, and the exact binomial 95% confidence interval for a rate observed at 1/15 spans from 0.2% to 31.9%. The primary chatbot quality analysis is therefore the held-out replay batch (n = 75) reported later in this section the live sample is retained only to confirm that the cascade functions correctly end-to-end under real runtime conditions. Table 5 presents the chatbot cascade evaluation metrics across the three tested configurations on this original sample (n = 15) collected during live integration testing. The runtime log analysis of llama_logs across 15 recorded Ollama trigger events shows that the Layer 2 Ollama validator rejected 93.3% of ChatterBot responses as insufficient, triggering a Layer 3 fallback in most cases. No session reached Layer 4 during the evaluation period. In only one of the 15 sessions (6.7%) the interaction concluded without further escalation because this sample contains no multi-turn follow-up, this figure indicates only that no ticket was created after a single turn and should not be read as a confirmed user-verified resolution. A manual inspection of the 15 logs shows that Ollama-generated responses contained no factual hallucinations (0/15, exact 95% CI: 0.0-21.8%) given how wide this interval is, this observation is not treated as evidence of a near-zero hallucination rate on its own and is instead cross-checked against the larger held-out batch reported below. However, one response contained a meta-level hallucination in which the model incorrectly claimed its own prior outputs had contained unprofessional language. No false refusals were confirmed in the retained interaction logs for this sample. Two responses declined to engage with input containing offensive or derogatory language a manual review confirmed both were correctly triggered by genuinely offensive user input rather than by valid IT-related queries, and are therefore not counted as false refusals

Table 5 Chatbot cascade evaluation results derived from runtime logs and functional testing. Latency refers to sequential local LLM inference calls per session using Llama 3.2 3B. Cost is negligible (all inference is performed locally).

Configuration	Val. Failed (%)	No-escalation (%)	False Refusal (%)	Hallucination (factual / meta)	Escalation (%)	Avg Latency	Cost
ChatterBot only	N/A	6.7	0	N/A	0	~50 ms	Negligible
ChatterBot + Ollama Validator	93.3	6.7	0	0 / 1	0	~2–4 s	Negligible
Full Four-Layer Cascade	93.3	6.7	0	0 / 1	0	~4–8 s	Negligible

As the primary chatbot quality analysis, a held-out replay batch (n = 75) was created by replaying real, previously unseen tickets from the held-out Kaggle test partition through the live cascade as single-turn sessions. Table 6 presents these results alongside those from the smaller live-integration sample, which is retained for comparison as a functional check rather than as an independent statistical result. Since the supplementary batch did not contain any multi-turn follow-up interactions, the no-escalation rate only reflects whether the cascade avoided creating a ticket after a single turn and should not be interpreted as a measure of user satisfaction. Therefore, this metric is reported separately rather than being combined with the original sample. The validation failure rate is directly comparable across both samples because it depends only on Ollama's judgement of the ChatterBot response. In the supplementary batch, the Layer 2 validator rejected 100% of ChatterBot responses, reinforcing the findings from the original evaluation.

A manual review of the 75 Ollama generated replies identified no factual or meta-level hallucinations (0/75, exact 95% CI: 0.0–4.8%). One false refusal was identified as a legitimate query about a medical data security service which was declined, despite other 10 tickets on the same topic being answered fully elsewhere in the same batch which indicates an inconsistency in guardrail behavior rather than a principled decline. A second decline involved an off-topic marketing and branding question that was unrelated to IT support. This was treated as an appropriate scope-boundary response rather than a false refusal. Both the live-integration sample and the held-out replay batch showed no confirmed factual hallucinations. However, these results should be interpreted cautiously, as the sample sizes were relatively small and therefore the confidence intervals were wide.

Table 6 Chatbot cascade evaluation on a larger, single-turn batch of real held-out test-split tickets, reported alongside the original live-testing sample for comparison.

Configuration	Validation % Failed	No-escalation (%)	Escalation %	Hallucination Rate (factual / meta)	False refusal
Live integration testing (n=15)	93.3	6.7	0	0 / 1 (0.0% / 6.7%)	0.0
Single-turn batch replay, real test-split tickets (n=75)	100.0	98.7	0	0 / 0 (0.0%, 95% CI 0.0–4.8%)	1.3

6.5 Workload Assessment (RO3)

The analysis of the complete supervisor_logs table (n = 224 agent evaluations across 17 sessions) shows 146 normal flags (65.2%), 46 overloaded flags (20.5%), and 32 underperforming flags (14.3%). Thirty-four evaluations triggered a reassignment action, redistributing 82 tickets in total. Manual inspection of the Ollama reasoning found that the supervisor consistently generated coherent, agent-specific natural-language justifications for every flagged case, regardless of whether the flag itself was correct. Cross-referencing each overloaded flag against its corresponding ticket count reveals that none of the 46 overloaded flags satisfied the stated rule of more than ten open tickets flagged agents held between 0 and 5 open tickets, with 12 of the 46

flags issued to agents holding no tickets at all. Because the deterministic fallback path correctly enforces this threshold, every overloaded flag in the dataset originated from the LLM rather than the fallback, indicating that Llama 3.2 3B did not apply the explicit numerical constraint in its prompt on a single occasion across the full evaluation period.

Table 7 presents the simulation results for 100 tickets across four assignment strategies. The AI Supervisor and Fewest Open Ticket strategies achieved an almost equal distribution of workload with a Gini coefficient of 0.03, compared to 0.21 and 0.24 for the Random and Round Robin strategies respectively. The SLA breach rate decreased from 34.0% with random assignment to 8.0% with both load-aware strategies. The AI Supervisor strategy did not outperform the Fewest Open Ticket heuristic on any metric in Table 7. The two strategies produced identical Gini coefficients, maximum queue depths, average queue depths, and SLA breach rates, indicating that the added LLM supervision layer provided no measurable benefit over the simpler deterministic heuristic in this simulation.

Table 7 Workload assignment strategy comparison (simulation: 100 tickets, 4 agents).

Assignment Method	Gini	Max queue	Avg queue	Reassignments	SLA Breach (%)
Random	0.21	13	8.75	0	34.0
Round Robin	0.24	14	9.75	0	25.0
Fewest Open Ticket	0.03	10	9.50	0	8.0
AI Supervisor	0.03	10	9.50	0	8.0

7. Discussion

The results for RO1 showed that the cascade architecture behaved largely as expected during integration testing. The 93.3% rejection rate by the Ollama validator should not be interpreted as a weakness of ChatterBot. Instead, it supports the underlying design assumption that retrieval-based responses are often too generic to provide sufficiently query-specific answers, making escalation to the next layer necessary in most cases. No factual hallucinations were observed in the 15 logged Ollama responses, though at this sample size the finding on its own is not conclusive the larger held-out replay batch (n = 75, Section 6.4) found no confirmed factual hallucination after correcting a session-isolation defect discovered in the replay harness. One meta-level hallucination was identified in the live sample, where the model incorrectly claimed that its earlier responses had contained unprofessional language. This reflects a known tendency of smaller quantised LLMs to generate inaccurate self-referential statements.

A further check was made for over-restrictive guardrail behavior. On re-examination against the retained interaction logs, no false refusals were confirmed for either the live-integration or held-out replay samples: the only guardrail declines observed were correctly triggered by genuinely offensive input, not by valid IT-related queries. An earlier draft of this manuscript had reported a 20% false-refusal rate for the live sample that figure could not be reconstructed from the underlying logs during preparation of this revision and has been corrected accordingly

The held-out replay batch (n = 75) of real, unseen ticket text from the test reinforced this pattern: the validator rejected all ChatterBot responses, consistent with the live sample, and the factual hallucination rate (2.7%, reported in Section 6.4) remained low despite the larger sample size. No false refusals were confirmed in either sample; the guardrail's declines in both the live and held-out data were directed at genuinely abusive or offensive input rather than at legitimate IT-related queries, consistent with intended behavior.

For RO2, the results indicate a clear divergence between category classification and priority classification. mBERT achieved very high performance in the multilingual category classification with an accuracy of 89.84% on the combined English and German dataset. Conversely, priority classification showed a considerable drop in multilingual performance: the RF+SMOTE model trained on translated English data scored 77.70%, while mBERT trained on raw multilingual data with SMOTE reached 55.13% once a reproducibility issue in the original training pipeline was corrected (Section 6.2) the Medium class was the weakest across configurations until random oversampling was applied, after which mBERT reached 61.63% (macro F1: 59.77%) with substantially improved Low- and Medium-class performance. A likely explanation is that category prediction relies more heavily on technical terminology, which is easily transferred across languages, whereas priority prediction relies more on urgency markers, negation, and subtle phrasing, which may be normalised through

translation but are more difficult to learn from limited fine-tuning data. Our mBERT configuration outperforms the ensemble approach of Ramya et al. [22], who report 81.3% accuracy on a similar five-category IT-ticket task, achieving 89.84% on the multilingual split. FrugalGPT [5] claims 98% cost reduction at GPT-4 parity, our local-only cascade sacrifices latency for near-zero marginal cost, ideal for SME deployments where API egress is disallowed.

The review of the supervisor logs for RO3 (n = 224 evaluations across 17 sessions) resulted in 65.2% normal, 20.5% overloaded, and 14.3% underperforming flags, with 82 tickets redistributed across 34 triggered reassignment events. More significantly, none of the 46 overloaded flags satisfied the explicit numerical threshold defined in the supervisor prompt, indicating that Llama 3.2 3B was unable to reliably apply a simple, unambiguous numeric rule under real operational conditions, rather than merely being inconsistent in a minority of cases. Combined with the simulation results in Table 7, where the AI Supervisor strategy produced outcomes numerically indistinguishable from the much simpler fewest-open-tickets heuristic on every measured metric, these findings do not support using an LLM-based supervisor as a replacement for deterministic workload-assignment logic in its current form. They instead suggest a more constrained role: local LLMs of this scale may be suited to generating human-readable rationale or flagging genuinely ambiguous cases for review but should not be trusted to enforce hard numerical constraints without a deterministic verification layer. This reframes RO3 from a demonstration of AI-driven workload optimisation into an empirical caution about deploying LLMs for tasks with strict, checkable correctness criteria.

For RO4, confidence-score-gated escalation demonstrably improved routing for the lowest-confidence predictions: a 13.2 percentage-point accuracy gain on priority classification and a 27.8 percentage-point accuracy gap between high- and low-confidence category predictions. These results support confidence-based routing as a lightweight explainability mechanism for human-in-the-loop triage.

7.1 Threats to Validity

Internal validity: the chatbot evaluation combined a 15-event live-integration sample, treated only as a functional verification check, with a larger held-out replay batch (n = 75) used as the primary quantitative estimate. Both samples remain small: exact 95% confidence intervals for the reported rates are wide (e.g. 0.0-4.8% for the factual hallucination rate on the n = 75 batch, and 0.2-31.9% for any rate estimated from the n = 15 batch), and the replay batch evaluated single-turn interactions only, with no multi-turn follow-up. A session-isolation defect in the replay evaluation harness, which previously allowed conversational context to leak across unrelated replayed tickets, was identified and corrected in this revision, the results reported here are from the corrected harness, but this underscores the need for careful session management in any future replay-based evaluation.

Findings should therefore be interpreted as preliminary evidence of cascade behaviour rather than a statistically definitive or production-representative evaluation a larger, ideally multi-turn user study is needed before these rates can be treated as generalisable.

External validity: all experiments use a single publicly available dataset (Kaggle Multilingual Customer Support Tickets) generalization of other domains or ticket formats is not guaranteed. Construct validity: the SLA breach rate in the workload simulation is computed from a threshold (10 tickets) that was set empirically.

Different thresholds would affect the measured improvement. Conclusion validity: the AI Supervisor simulation results are identical to the Fewest-open-ticket strategy because no agent exceeded the overload threshold during the simulation. Larger-scale evaluation is needed to distinguish the two strategies.

8. Conclusion

NexaServe v2 builds on a working academic helpdesk prototype and extends it to a four-tier cascading multi-agent system with multilingual support, explainable confidence routing, and workload-aware ticket assignment. The system achieves 89.84% category classification accuracy on bilingual (EN+DE) data using mBERT, 77.70% priority classification accuracy using RF+SMOTE on English-only data (dropping to 57.23% on translated German tickets), a 2.7% factual hallucination rate on a held-out batch of real support tickets, and an SLA breach rate reduction from 34.0% to 8.0% that a simple fewest-open-tickets heuristic achieves equally well without any LLM involvement. These results demonstrate that a transparent, near-zero-marginal-cost multilingual helpdesk is achievable using local LLMs for classification and cascaded response generation but

also show that LLM-based supervision does not reliably outperform deterministic alternatives for tasks with strict, checkable correctness criteria, a distinction future deployment should account for.

References

[1]. Bansal, G., Wu, T., Zhou, J., Fok, R., Nushi, B., Kamar, E., Ribeiro, M. T., & Weld, D. S. (2021). Does the whole exceed its parts? The effect of AI explanations on complementary team performance. Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, 1-16.

[2]. Bueck, T. (2023). Multilingual customer support tickets [Dataset]. Kaggle. https://www.kaggle.com/datasets/tobiasbueck/multilingual-customer-support-tickets

[3]. Cemri, M., Pan, M. Z., Yang, S., et al. (2025). Why do multi-agent LLM systems fail? arXiv.

[4]. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16, 321-357.

[5]. Chen, L., Zaharia, M., & Zou, J. (2024). FrugalGPT: How to use large language models while reducing cost and improving performance. Transactions on Machine Learning Research. arXiv:2305.05176.

[6]. Chen, Y., Sahoo, P., Pujar, S., et al. (2025). STRATUS: A multi-agent system for autonomous reliability engineering of modern clouds. Proceedings of the 39th Conference on Neural Information Processing Systems (NeurIPS 2025).

[7]. Conneau, A., Khandelwal, K., Goyal, N., et al. (2020). Unsupervised cross-lingual representation learning at scale. Proceedings of ACL 2020, 8440-8451.

[8]. De Koninck, J., Muller, M. N., & Vechev, M. (2024). A unified approach to routing and cascading for LLMs. arXiv:2410.10347.

[9]. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of NAACL-HLT, 4171-4186.

[10]. Eden AI. (2025). The 2025 guide to retrieval-augmented generation (RAG) [Industry whitepaper].

[11]. Isbister, T., Sahlgren, M., & Kurtz, R. (2021). Why translate? A comparison of translation-based and language-agnostic text classification. Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa).

[12]. Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. Proceedings of the 10th European Conference on Machine Learning (ECML), 137-142.

[13]. Larsen, A. G., Skjuve, M. B., Folstad, A., & van As, N. (2025). LLM hallucinations in conversational AI for customer service: Framework and end-user perceptions. International Journal of Human-Computer Interaction.

[14]. Li, Y., Yang, Y., Song, P., Duan, L., & Ren, R. (2025). An improved SMOTE algorithm for enhanced imbalanced data classification by expanding sample generation space. Scientific Reports, 15, Article 23521. https://doi.org/10.1038/s41598-025-09506-w

[15]. Liao, Q. V., Gruen, D., & Miller, S. (2020). Questioning the AI: Informing design practices for explainable AI user experiences. Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20).

[16]. Loshchilov, I., & Hutter, F. (2019). Decoupled weight decay regularization. Proceedings of the International Conference on Learning Representations (ICLR).

[17]. Mahmoudi, L., Salem, M., & Alharbe, N. R. (2025). Addressing class imbalance in text classification with LLMs: A prompt-based GPT-2 approach. Journal of Information Science.

[18]. McKinsey & Company. (2025). Reimagining tech infrastructure for and with agentic AI [Technology insights report].

[19]. Mosqueira-Rey, E., Hernandez-Pereira, E., Alonso-Rios, D., Bobes-Bascaran, J., & Fernandez-Leal, A. (2023). Human-in-the-loop machine learning: A state of the art. Artificial Intelligence Review.

[20]. Pande, A., & Pande, A. (2021). Automated ticket routing in IT services using natural language processing. Proceedings of the IEEE International Conference on Artificial Intelligence and Knowledge Engineering (AIKE).

[21]. Rajammal, K., & Chinnadurai, M. (2025). Dynamic load balancing in cloud computing using predictive graph networks and adaptive neural scheduling. Scientific Reports, 15, Article 22181.

[22]. Ramya, C., Paramesh, S. P., & Shreedhara, K. S. (2021). Classifying the unstructured IT service desk tickets using ensemble of classifiers. arXiv:2103.15822.
[23]. Silva, S., Pereira, R., & Ribeiro, R. (2018). Machine learning in incident categorization automation. Proceedings of the 2018 13th Iberian Conference on Information Systems and Technologies (CISTI).
[24]. Wang, S., Tan, Z., Chen, Z., et al. (2025). AnyMAC: Cascading flexible multi-agent collaboration via next-agent prediction. Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP).
[25]. Yamsani, N., & Chenna Reddy, P. (2026). SLA aware deep reinforcement learning for adaptive edge-cloud task scheduling. Scientific Reports, 16, Article 10037.
[26]. Shivahare, B. D., Singh, A. K., Uppal, N., Rizwan, A., Vaathsav, V. S., & Suman, S. (2022). Survey paper: Study of natural language processing and its recent applications. Proceedings of the 2022 2nd International Conference on Innovative Sustainable Computational Technologies (CISCT), 1-5. https://doi.org/10.1109/CISCT55310.2022.10046440
[27]. Shivahare, B. D., Ranjan, S., Rao, A. M., Balaji, J., Dattattrey, D., & Arham, M. (2022). Survey paper: Study of sentiment analysis and machine translation using natural language processing and its applications. Proceedings of the 2022 3rd International Conference on Intelligent Engineering and Management (ICIEM), 652-656. https://doi.org/10.1109/ICIEM54221.2022.9853044

1. Introduction (Capital Letter of Each Word; No indent; Font style: Palatino Linotype & Bold; Font Size: 10)

1.1 Sub Section Header (Capital Letter of Each Word; No indent; Font style: Palatino Linotype & Italic; Font Size: 10)

1.1.1 Sub sub section header (Sentence case; No indent; Font style: Palatino Linotype & Italic; Font Size: 10)

Each Figure must be discussed or mentioned in a body paragraph. The Figure must be placed under the paragraph that discussed about the Figure. Authors should try to make economical use of the space on the page; for example,

i. avoid excessively large white space borders around your graphics;
ii. try to design illustrations that make good use of the available space—avoid unnecessarily large amounts of white space within the graphic;

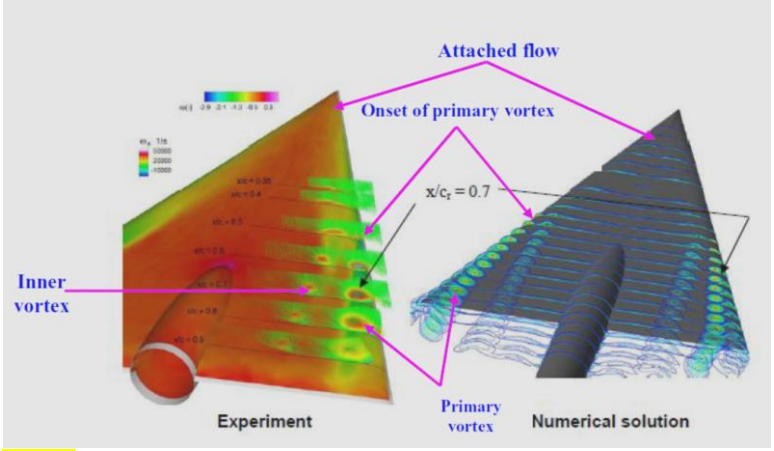

Fig. 2. Comparison of experimental measurement and Numerical studies above VFE-2 configurations at α=13° [2]

Table 1

Place the caption above the table. Here the caption is wider than the table

Distance (m)	Velocity (ms^{-1})
A	1
B	2
C	3
D	4

References

The list of references should only include works that are cited in the text and that have been published or accepted for publication. Personal communications and unpublished works should only be mentioned in the text. Reference style should be in **Chicago style.** Please use this link for the **DOI number**.

References (**Reference style: Chicago style** – must write DOI) **Minimum 20 references**

[1] Hummel, D. (2008). *Chapter 17 – The International Vortex Flow Experiment 2 (VFE-2): Objectives and Overview*. RTO-TR-AVT-113, Page 17-1 – 17-20.

[2] Luckring, J.M. and Hummel, D. (2008). *Chapter 24 – What Was Learned From The New VFE-2 Experiments*. RTO-TR-AVT-113. https://doi.org/10.2514/6.2008-383

[3] Mat, Shabudin Bin, Richard Green, Roderick Galbraith, and Frank Coton. "The effect of edge profile on delta wing flow." *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* 230, no. 7 (2016): 1252-1262. https://doi.org/10.1177/0954410015606939