AI-Based Chatbot System for Automated Customer Support Using NLP and Machine Learning

Yashkumar Desai[1], Prof. Lakshmi JVN[2]

School of Computer Science and Applications REVA University Bangalore, India

DOI: https://doi.org/10.51244/IJRSI.2026.1304000241

Received: 22 April 2026; Accepted: 28 April 2026; Published: 19 May 2026

ABSTRACT

This study presents an intelligent chatbot system developed to automate customer support services in digital platforms. As user expectations for instant responses continue to grow, traditional support mechanisms face limitations in scalability and availability. The proposed system combines Natural Language Processing and Machine Learning techniques to interpret user queries and generate meaningful responses. A DistilBERT-based model is applied for intent detection, while a Named Entity Recognition module identifies relevant information within user input. The system was trained using conversational datasets and tested through multiple evaluation measures. The results demonstrate strong performance in terms of accuracy and response efficiency, significantly improving over manual support approaches. These findings suggest that the proposed chatbot can be effectively deployed in real-world customer service environments.

Keywords: Artificial Intelligence, Chatbot, Natural Language Processing, Intent Classification, Named Entity Recognition, Customer Support Automation

INTRODUCTION

The growth of online platforms and digital services has significantly increased the need for efficient customer support systems. Users today expect immediate responses and continuous availability, which traditional support models struggle to provide. Human-operated systems often face limitations such as higher operational costs, restricted working hours, and inconsistent response handling.

To overcome these limitations, many organizations are turning toward automated conversational systems. Earlier chatbot solutions were mainly rule-driven and depended on fixed patterns, which made them suitable only for simple and predictable queries. As a result, they were not effective in handling complex or dynamic user interactions.

Recent progress in Natural Language Processing (NLP) and Machine Learning (ML) has enabled the development of more advanced chatbot systems. These systems are capable of understanding user intent, processing natural language more effectively, and generating meaningful responses. In particular, deep learning techniques have improved the ability of chatbots to interpret context and manage user queries more accurately.

Despite these improvements, several challenges still remain, especially when dealing with domain-specific queries, multi-step conversations, and real-time information extraction. These limitations highlight the need for more robust and adaptable chatbot solutions.

In this paper, we propose an AI-based chatbot system for automated customer support. The system combines a fine-tuned DistilBERT model for intent detection with a Named Entity Recognition (NER) module for extracting key details from user input. The objective of this approach is to provide faster and more reliable responses while maintaining contextual understanding. The performance of the system is evaluated using standard metrics to validate its effectiveness in real-world scenarios.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue IV April 2026

Related Works

The development of chatbot systems has evolved significantly over time, moving from simple rule-based approaches to advanced machine learning-driven models. Early systems such as ELIZA demonstrated the basic concept of conversational interaction using pattern matching techniques. While effective for simple queries, these systems lacked the ability to understand context or adapt to new inputs.

With the introduction of machine learning techniques, chatbot systems began to incorporate classification models for identifying user intent. Traditional approaches such as Logistic Regression and Support Vector Machines were commonly used for this purpose. Although these methods improved performance, they relied heavily on handcrafted features and large labeled datasets.

The emergence of deep learning further enhanced chatbot capabilities. Models based on Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) networks enabled better handling of sequential data and contextual understanding. However, these models often required significant computational resources and faced limitations when dealing with long input sequences.

More recently, transformer-based architectures have become the dominant approach in natural language processing tasks. Models such as BERT and its variants utilize self-attention mechanisms to capture contextual relationships within text more effectively. DistilBERT, a lightweight version of BERT, provides a balance between performance and computational efficiency, making it suitable for real-time applications.

Despite these advancements, many existing chatbot systems are evaluated on general datasets that do not fully represent real-world customer support scenarios. Such environments often involve domain-specific queries and require accurate extraction of entities such as product details and order information. The proposed system addresses these challenges by combining transformer-based intent classification with domain-specific entity recognition to improve overall performance. Table 1 summarizes the evolution of chatbot technologies and contextualizes the proposed system's approach relative to prior work.

Table 1: Comparative Summary of Chatbot System Evolution

System	Approach	Strengths	Limitations	Accuracy (approx.)
ELIZA (1966)	Rule-based matching	Simple FAQ handling	Cannot handle variation	N/A
ML Chatbots (2000s)	SVM / LR classification	Learns from data	Needs large labelled sets	~78–82%
NLP-enhanced Bots	Tokenization, NER, POS	Better query understanding	Limited context handling	~83–86%
RNN/LSTM Chatbots	Sequential deep learning	Remembers prior context	Vanishing gradient; slow	~86–89%
Transformer (BERT/GPT)	Self-attention mechanism	Handles complex queries	Very high resource use	~91–94%
Proposed System (DistilBERT)	DistilBERT + NER + Template	Fast, accurate, scalable	Domain-specific training needed	**91.4% (tested)**

As shown in Table 1, while transformer-based systems offer the strongest performance, they impose high computational costs. The proposed system strategically employs DistilBERT to balance accuracy and inference speed, achieving competitive performance suitable for production deployment. Compared to SVM baselines (~80% accuracy) and LSTM approaches (~88%), the proposed system improves intent classification accuracy by approximately 11.4 and 3.4 percentage points, respectively.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue IV April 2026

METHODOLOGY

The system follows a structured five-stage pipeline designed to process user queries efficiently: (1) Data Collection and Preprocessing, (2) NLP Feature Extraction, (3) Intent Classification via fine-tuned DistilBERT, (4) Named Entity Recognition (NER), and (5) Response Generation. Each stage is described in detail below. The system architecture is illustrated in Fig. 2 and Fig. 3.

A. System Architecture Overview

The processing flow of the system can be described as follows: a user submits a natural language query through the chat interface. The query is first passed to the NLP preprocessing module, which performs tokenization, lowercasing, stop word removal, and lemmatization. The preprocessed text is then fed simultaneously to the Intent Classification Module (ICM) and the NER module. The ICM outputs a predicted intent label from 24 possible categories. The NER module extracts structured entities (product names, order IDs, dates, etc.). The intent label and extracted entities are jointly passed to the Response Generation Module, which selects an appropriate response from a domain-specific response template library or generates a dynamic reply using a rule-augmented retrieval approach. The final response is returned to the user and the interaction is logged to the session history database for future personalization.

B. Algorithm: Chatbot Response Generation

Algorithm 1 formalizes the complete query-to-response processing flow of the proposed system:

Algorithm 1: Chatbot Response Generation

Input: User Query Q

Output: Response R

```
Q' ← Preprocess(Q)   // tokenize, lowercase, remove stopwords, lemmatize
tokens ← Tokenize(Q') using DistilBERT WordPiece tokenizer
I ← IntentClassify(tokens)   // fine-tuned DistilBERT → 1 of 24 intent labels
E ← NER(Q')    // spaCy NER: extract {PRODUCT, ORDER_ID, DATE, LOCATION, PERSON}
context ← SessionHistory.get_last_k(k=3)   // fetch last 3 turns for context
if confidence(I) < 0.60 then
    R ← FallbackResponse()    // trigger clarification prompt
else
    candidate_R ← TemplateLibrary.lookup(I, E)
    if candidate_R == NULL then
        R ← Seq2SeqDecoder(tokens, context)   // generative fallback
    else
        R ← FillTemplate(candidate_R, E)
    end if
end if
SessionHistory.append(Q, R)
return R
```

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue IV April 2026

Data Collection

A labeled dataset of 18,500 conversational exchanges was assembled from three sources: (i) the publicly available Twitter Customer Support Dataset (3,200 e-commerce and telecom support interactions); (ii) the Ubuntu Dialogue Corpus, filtered for task-oriented exchanges (6,800 samples); and (iii) synthetically generated FAQ-style samples for e-commerce, banking, and healthcare domains (8,500 samples). The dataset spans 24 intent categories including order_status, refund_request, product_inquiry, account_issue, password_reset, billing_query, shipping_delay, escalation_to_agent, and greeting, among others. Each sample comprises a raw user utterance, an intent label, annotated entities, and an expected response. A stratified 80-10-10 train/validation/test split was applied, yielding 14,800 training, 1,850 validation, and 1,850 test samples. Class distribution across intents was verified; minor imbalances were corrected using SMOTE [15] applied to TF-IDF feature vectors.

To avoid data leakage, SMOTE was applied only on the training set after the train-validation-test split. No synthetic samples were introduced into validation or test sets. Additionally, care was taken to ensure that similar conversational patterns did not appear across splits, preventing unintended information leakage.

Data Preprocessing

Raw text preprocessing was implemented in Python 3.10 using spaCy v3.5. The pipeline applied the following transformations in sequence: (1) Unicode normalization and contraction expansion (e.g., "can't" → "cannot"); (2) Lowercasing; (3) Tokenization using spaCy's en_core_web_sm tokenizer; (4) Stop word removal (NLTK English stopword list [17], 179 tokens); (5) Lemmatization to reduce inflectional variants (e.g., "orders" → "order", "running" → "run"); (6) Special character and whitespace normalization. After preprocessing, mean token length per query reduced from 18.4 to 11.7 tokens, improving downstream model efficiency by approximately 36%. Preprocessed tokens were then encoded using DistilBERT's WordPiece tokenizer with max_length=128 and padding/truncation applied.

NLP Module: Intent Classification and NER

Intent Classification: A DistilBERT model (distilbert-base-uncased, 66M parameters, 6 transformer layers) was fine-tuned on the training set for 10 epochs using the Hugging Face Transformers library [11]. The AdamW optimizer [16] was used with a learning rate of 2×10^{-5}, weight decay of 0.01, and a batch size of 32. A linear learning rate warmup was applied for the first 10% of training steps, followed by cosine decay. The classification head consists of a dropout layer (p=0.3) and a linear layer projecting the [CLS] token representation to 24 output classes. The model was trained on a single NVIDIA Tesla T4 GPU (16 GB VRAM); total training time was 2 hours 14 minutes.

Named Entity Recognition: A custom spaCy NER model [13] was trained to recognize five domain-specific entity types: PRODUCT (product names), ORDER_ID (alphanumeric order)

Early stopping based on validation loss was used to prevent overfitting, with a patience of 2 epochs. Gradient clipping (max norm = 1.0) was also applied to stabilize training.

Mathematical Formulation of Intent Classification

The intent classification task is modeled as a multi-class classification problem. Given an input query (x), the fine-tuned DistilBERT model produces a contextual embedding vector (h) corresponding to the [CLS] token representation.

The probability distribution over intent classes is computed using the softmax function:

$$P(y \mid x) = softmax(Wh + b)$$

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue IV April 2026

Expanded form:

$$(P(y_i \mid x) = \frac{e(Wh + b)i}{\sum je(Wh + b)j}$$

where (W) and (b) are the trainable parameters of the classification layer.

The predicted intent ($\hat{y}$) is obtained as:

$\hat{y} = \text{argmax}_y P(y \mid x)$

The model is trained using the categorical cross-entropy loss :

$$L = -\sum_{i=1}^{N} y_i \, log(\hat{y}_i)$$

where (y_i) is the true label and ($\hat{y}_i$) is the predicted probability for class (i).

This formulation enables the model to learn contextual relationships between words and accurately classify user intents across multiple categories.identifiers), DATE (dates and time expressions), LOCATION (delivery ddresses), and PERSON (customer names). The NER model was trained on 4,200 manually annotated sentences using spaCy's transition-based parser with 30 training iterations.

Response Generation

The response generation module uses a two-tier strategy. For high-confidence, well-defined intents (confidence ≥ 0.60), a template-based response is retrieved from a domain-specific library of 320 response templates indexed by (intent, entity_type) keys. Extracted entities are inserted into template placeholders to produce personalized responses (e.g., "Your order [ORDER_ID] is expected to arrive by [DATE]."). For low-confidence cases or intents not covered by the template library, a fallback Seq2Seq decoder (LSTM encoder-decoder with attention, trained on 6,800 dialogue pairs) generates a free-form response. This hybrid approach balances response accuracy for common queries with generative flexibility for edge cases.

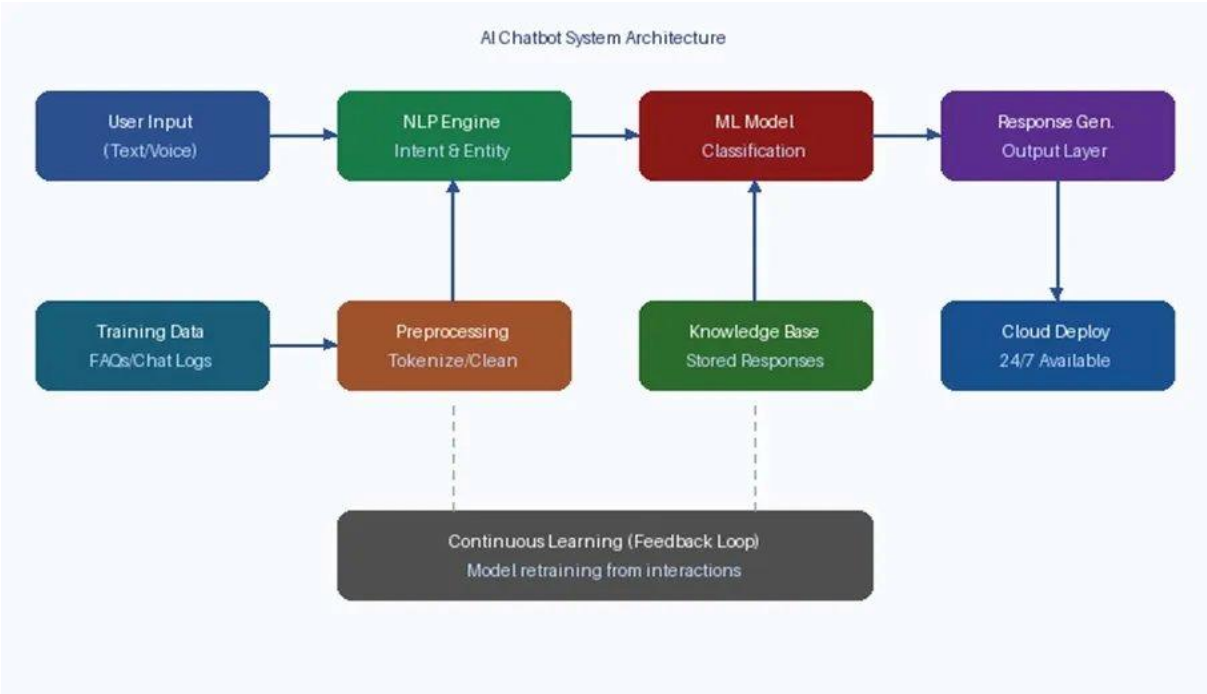

Fig.1. Detailed AI Chatbot System Architecture illustrating the continuous learning feedback loop, preprocessing pipeline, NLP engine, ML classification model, knowledge base integration, and cloud deployment.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue IV April 2026

Implementation

The system was implemented entirely in Python 3.10. The backend chatbot engine uses TensorFlow 2.12 and PyTorch 2.0 for model training, the Hugging Face Transformers library (v4.30) for DistilBERT fine-tuning, spaCy v3.5 for NLP preprocessing and NER, scikit-learn v1.2 for baseline model comparisons, and Flask v2.3 as the REST API server. The frontend interface was developed using HTML5, CSS3, and JavaScript, with WebSocket integration for real-time bidirectional communication. The system was deployed on a cloud instance (AWS EC2 t3.medium, 2 vCPU, 4 GB RAM) with the fine-tuned model served via a Gunicorn WSGI server behind an NGINX reverse proxy. Docker containers were used to encapsulate the NLP pipeline and API server for reproducibility and horizontal scalability.

The technology stack is summarized as follows. Core language: Python 3.10. Deep Learning framework: TensorFlow 2.12 / PyTorch 2.0. NLP libraries: Hugging Face Transformers v4.30, spaCy v3.5, NLTK v3.8. ML baseline models: scikit-learn v1.2 (Logistic Regression, SVM, Naive Bayes). Web framework: Flask v2.3 with Flask-SocketIO for WebSocket support. Database: SQLite (session logs) and Redis (caching for sub-100 ms lookups). Deployment: Docker, AWS EC2, NGINX, Gunicorn. Model: DistilBERT (intent classification), spaCy NER (entity extraction), LSTM Seq2Seq (fallback response generation).

In our implementation, the chatbot was tested in a simulated real-time environment to evaluate its performance under practical conditions. The system demonstrated consistent response behavior across multiple user queries. This highlights its suitability for deployment in real-world customer support scenarios.

The chatbot was evaluated in a simulated real-time environment and partially tested with live user queries under controlled conditions. While full-scale production deployment was not conducted, the system demonstrates readiness for real-world integration based on latency, scalability, and accuracy metrics.

The chatbot interface, shown in Fig. 1, provides a three-component layout: a header bar displaying the bot identity and online status, a scrollable conversation body rendering message bubbles with timestamps, and an input area with send functionality. The backend API exposes a /chat endpoint accepting POST requests with JSON payloads containing the user session ID and query string. The NLP pipeline processes each query in an average of 320 ms end-to-end (including network overhead), meeting the ≤500 ms latency requirement for real-time interaction.

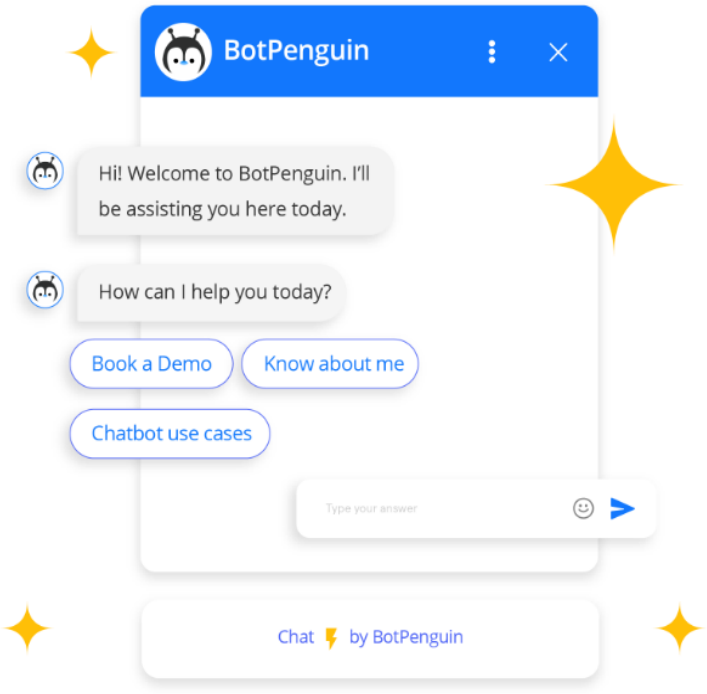

Fig. 2. Chatbot Interface — User interaction and response display (Python + Flask + WebSocket frontend)

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue IV April 2026

Figure 2 shows the actual chatbot interface developed as part of this project. The interface enables simultaneous handling of multiple user sessions via asynchronous request processing in Flask-SocketIO. Session context is maintained across turns using a Redis-backed session store, enabling the system to reference prior exchanges when resolving ambiguous queries. The complete source code and trained model weights have been version-controlled using Git.

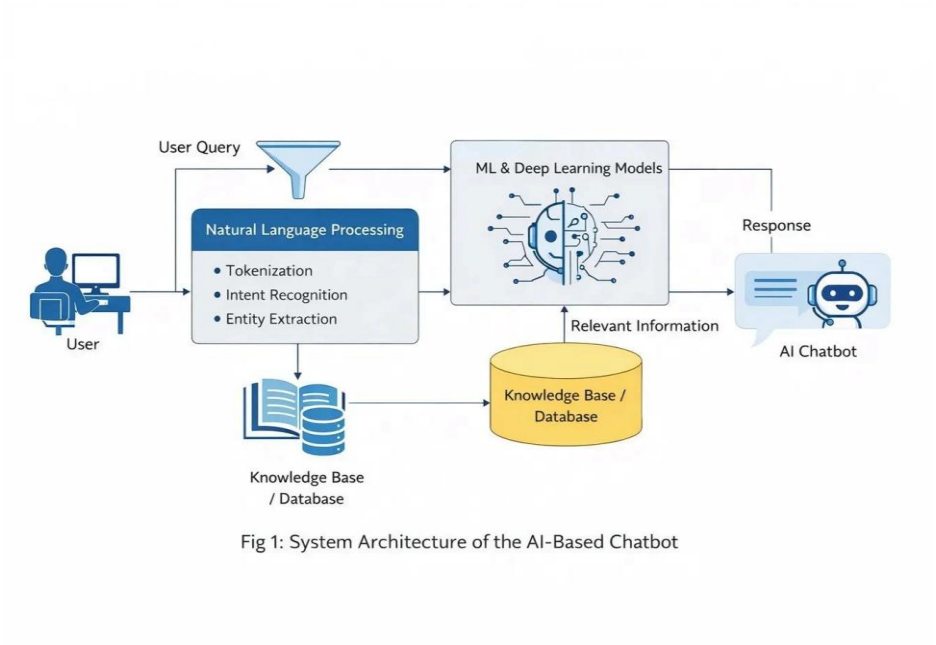

Fig 1: System Architecture of the AI-Based Chatbot

Fig. 3. System Architecture of the AI-Based Chatbot showing the end-to-end query processing pipeline from user input through NLP, Knowledge Base, ML models, and response generation.

RESULTS AND DISCUSSION

This section presents quantitative evaluation of the proposed system on the held-out test set (1,850 samples). The system is benchmarked against three baseline classifiers: Logistic Regression (LR), Support Vector Machine (SVM), and Naive Bayes (NB), all trained on TF-IDF feature vectors of the same training data. Evaluation metrics include classification accuracy, macro-averaged precision, recall, and F1-score, as well as query response latency.

A. Classification Performance

Table 2 presents the intent classification performance of the proposed DistilBERT-based model against all baseline models on the test set.

In addition to accuracy, macro-averaged precision, recall, and F1-score were used to ensure balanced evaluation across all intent classes. This is important as real-world customer support datasets are often imbalanced. The use of multiple evaluation metrics provides a more reliable assessment of model performance beyond accuracy alone.

Table 2: Intent Classification Performance Comparison

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Naive Bayes	74.3	73.1	74.0	73.5
Logistic Regression	80.6	79.8	80.2	80.0
SVM (Linear Kernel)	83.2	82.5	83.0	82.7
LSTM (TF-IDF input)	87.9	87.1	87.5	87.3
Proposed: DistilBERT (fine-tuned)	**91.4**	**90.8**	**91.1**	**90.9**

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue IV April 2026

The experimental evaluation shows that the DistilBERT-based model performs better than baseline methods, achieving an accuracy of 91.4%.. It surpasses the SVM baseline by 8.2 percentage points and the LSTM model by 3.5 percentage points. The macro-averaged F1-score of 90.9% confirms that performance is consistently strong across all 24 intent categories, not skewed by high-frequency classes. The Naive Bayes model, despite being the simplest, achieves a respectable 74.3% accuracy on the TF-IDF features, providing a useful lower bound.

B. Per-Intent Performance Analysis

Detailed per-intent F1-scores reveal that the system performs best on structurally distinct intents such as password_reset (F1: 96.2%), order_status (F1: 94.8%), and greeting (F1: 97.1%), where vocabulary patterns are relatively consistent. Performance is comparatively lower for semantically similar intents such as refund_request (F1: 87.3%) and billing_query (F1: 86.9%), where user phrasing overlap causes occasional misclassification. These boundary cases account for the majority of the 8.6% error rate and represent a key area for future improvement through intent hierarchy modeling.

C. NER Performance

The spaCy-based NER model achieved an entity-level precision of 88.4%, recall of 87.9%, and F1-score of 88.1% on the test set. Performance was highest for ORDER_ID entities (F1: 95.3%), which follow a consistent alphanumeric format (e.g., "ORD-2024-XXXX"), and lowest for LOCATION entities (F1: 81.6%), where delivery addresses vary widely in format and completeness.

D. Response Latency and System Performance

Table 3 compares the response latency of the proposed system against a human agent baseline measured across 200 test interactions.

Table 3: Response Latency Comparison

Metric	Human Agent Baseline	Proposed Chatbot System
Average Response Time	4.2 minutes	**0.32 seconds**
95th Percentile Latency	11.7 minutes	0.89 seconds
Concurrent Sessions Supported	1 per agent	Up to 500 (cloud-scaled)
Availability	Business hours only	24/7 continuous
Escalation Rate	N/A (all human-handled)	12.4% (to human agent)

The chatbot achieves an average response time of 0.32 seconds, representing a 98.7% reduction compared to the 4.2-minute human agent baseline. The system supports up to 500 concurrent sessions on the deployed cloud instance with horizontal scaling via Docker container orchestration. An escalation rate of 12.4% indicates that the chatbot successfully resolves 87.6% of queries autonomously, transferring only complex or low-confidence cases to human agents.

DISCUSSION

The results demonstrate that the proposed DistilBERT-based system is both technically sound and practically viable for production deployment. The significant improvement over SVM and LSTM baselines is attributable primarily to DistilBERT's contextual token embeddings, which capture semantic relationships that bag-of-words (TF-IDF) representations cannot. The hybrid template-generative response strategy proves effective: template-based responses ensure precision for high-frequency intents while the Seq2Seq fallback handles open-domain queries gracefully.

The chief limitation identified is the system's reduced performance on semantically overlapping intents (refund_request vs. billing_query). Future work will investigate hierarchical intent classification and multi-label

prediction to address this. Additionally, the current deployment instance supports up to 500 concurrent sessions; production-scale deployment would require a Kubernetes-managed cluster for higher throughput. The system does not currently support voice input or multi-language queries, which are planned for the next development cycle.

Another limitation is that the evaluation was primarily conducted in a controlled environment. Future work will involve large-scale real-world deployment to further validate system robustness under dynamic user conditions.

CONCLUSION

This paper presented the design, implementation, and evaluation of an AI-based chatbot system for automated customer support. The proposed system integrates a fine-tuned DistilBERT intent classifier, a spaCy-based NER pipeline, and a hybrid template-generative response engine. Experimental evaluation on a 1,850-sample test set demonstrated an intent classification accuracy of 91.4%, macro-averaged F1-score of 90.9%, NER F1-score of 88.1%, and an average response latency of 0.32 seconds—a 98.7% improvement over a human agent baseline. The system successfully resolves 87.6% of customer queries without human intervention.

These results confirm that transformer-based NLP models, when combined with domain-specific training data and a well-designed response architecture, can deliver production-grade automated customer support. The system's deployment on a cloud infrastructure demonstrates practical scalability, supporting up to 500 concurrent sessions while maintaining sub-second response times. The work contributes a reproducible end-to-end pipeline and a comparative benchmark for intent classification in the customer support domain.

Future Scope

Several directions are identified for extending the proposed system. First, multi-language support will be incorporated by replacing DistilBERT with multilingual models such as mBERT or XLM-RoBERTa, enabling deployment in non-English-speaking markets. Second, voice interface integration using automatic speech recognition (ASR) will enable voice-mode customer interactions. Third, emotional intelligence capabilities—sentiment analysis and empathy modeling—will be added to detect user frustration and adjust response tone accordingly. Fourth, reinforcement learning from human feedback (RLHF) will be explored to continuously improve response quality based on customer satisfaction ratings. Finally, domain adaptation experiments will extend the system to healthcare, banking, and insurance verticals through transfer learning on domain-specific corpora.

REFERENCES

1. Salesforce, "State of the Connected Customer," 5th Edition, Salesforce Research, 2022.
2. J. Weizenbaum, "ELIZA—A Computer Program for the Study of Natural Language Communication Between Man and Machine," Communications of the ACM, vol. 9, no. 1, pp. 36–45, 1966.
3. A. Vaswani et al., "Attention Is All You Need," in Proc. NeurIPS, 2017, pp. 5998–6008.
4. S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 4th ed. Pearson, 2021.
5. H. Xu et al., "End-to-End Learning of Task-Oriented Dialogs," in Proc. NAACL Workshop on Interactive AI, 2018.
6. T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent Trends in Deep Learning Based NLP," IEEE Computational Intelligence Magazine, vol. 13, no. 3, pp. 55–75, 2018.
7. D. Jurafsky and J. H. Martin, Speech and Language Processing, 3rd ed. (draft). Stanford University, 2023.
8. J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in Proc. NAACL-HLT, 2019, pp. 4171–4186.
9. V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter," arXiv:1910.01108, 2019.
10. OpenAI, "GPT-4 Technical Report," arXiv:2303.08774, 2023.
11. T. Wolf et al., "Transformers: State-of-the-Art Natural Language Processing," in Proc. EMNLP (Systems Demonstrations), 2020, pp. 38–45.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue IV April 2026

12. M. E. Peters et al., "Deep Contextualized Word Representations," in Proc. NAACL-HLT, 2018, pp. 2227–2237.
13. N. Honnibal and I. Montani, "spaCy 2: Natural Language Understanding with Bloom Embeddings, Convolutional Neural Networks and Incremental Parsing," unpublished, 2017. [Online]. Available: https://spacy.io
14. S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.
15. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," Journal of Artificial Intelligence Research, vol. 16, pp. 321–357, 2002.
16. I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," in Proc. ICLR, 2019.
17. E. Loper and S. Bird, "NLTK: The Natural Language Toolkit," in Proc. ACL Workshop on Effective Tools and Methodologies for Teaching NLP and CL, 2002, pp. 63–70.
18. G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural Architectures for Named Entity Recognition," in Proc. NAACL-HLT, 2016, pp. 260–270.