International Journal of Science and Research Archive
eISSN: 2582-8185
Cross Ref DOI: 10.30574/ijsra
Journal homepage: https://ijsra.net/

(RESEARCH ARTICLE)

Check for updates

# Design and deployment of NLP-driven conversational agents in Java for the Insurance Sector

Mohan Rao Pulugulla * and Komal

*Department of Accounting, Finance, Economics, and Decision Sciences, Western Illinois University, Macomb, IL, USA.*

International Journal of Science and Research Archive, 2025, 16(03), 1226-1239

Publication history: Received on 08 August 2025; revised on 14 September 2025; accepted on 16 September 2025

Article DOI: https://doi.org/10.30574/ijsra.2025.16.3.2602

## Abstract

This paper presents the design, development, and deployment of a Natural Language Processing (NLP)-driven conversational agent built in Java, specifically customized for the insurance sector. The growing demand for 24/7 customer support, coupled with the need to reduce operational costs and improve service delivery, has made intelligent chatbots a compelling solution in the industry. Leveraging open-source Java NLP libraries such as Apache OpenNLP and Stanford CoreNLP, the proposed agent performs intent recognition, entity extraction, and dynamic dialogue management to assist users with tasks including policy inquiries, claim filing, fraud alerts, and live agent handoffs. The system is designed using a modular, scalable architecture that integrates with existing insurance infrastructure and adheres to strict data privacy regulations such as the NDPR and GDPR. Performance evaluation showed over 93% intent accuracy and high user satisfaction, with notable improvements in customer interaction speed and task completion. The study also discusses implementation challenges and outlines future enhancements, such as multilingual support, voice integration, and transformer-based model adoption. This research offers a replicable framework for deploying secure, efficient, and adaptable conversational agents within highly regulated industries.

**Keywords:** Conversational AI; NLP; Chatbots; Java; Insurance Technology; OpenNLP; Policy Automation; Claim Processing; Data Privacy; Intelligent Agent; NDPR; Customer Service Automation

## 1. Introduction

In the era of digital transformation, the insurance industry has experienced a paradigm shift in customer engagement, operational efficiency, and service delivery models. Traditional interactions through face-to-face consultations and telephone calls are rapidly giving way to digital platforms that promise faster, more convenient, and more cost-effective services. Among these innovations, Natural Language Processing (NLP)-driven conversational agents commonly referred to as chatbots have emerged as a significant technological intervention in the customer service landscape (Adamopoulou and Moussiades, 2020).

Conversational agents utilize Artificial Intelligence (AI) and NLP to simulate human-like conversations and interact with users in a natural language format (Jain et al., 2018). These systems are designed to understand, process, and respond to user input in real time, providing answers to queries, guiding decision-making, and executing service-related tasks. In the insurance sector, the application of such agents has the potential to streamline processes such as policy management, claims filing, fraud reporting, and customer onboarding, thereby reducing human workload and enhancing the overall customer experience (McTear, 2020).

Java, as a programming language, continues to play a pivotal role in enterprise software development, especially in finance and insurance sectors where system reliability, security, and long-term maintainability are critical. Despite the

* Corresponding author: Mohan Rao Pulugulla

dominance of Python in AI research, Java offers robust frameworks and libraries such as Apache OpenNLP, Stanford CoreNLP, and Deeplearning4j, which make it suitable for developing and deploying scalable NLP applications (Kumar and Natarajan, 2022). Furthermore, Java's strong ecosystem, object-oriented architecture, and backward compatibility make it a preferred choice for organizations seeking to integrate NLP agents into legacy systems without significant infrastructure overhaul.

Although the integration of AI-powered chatbots has been widely explored in various sectors including retail, healthcare, and hospitality, the insurance industry poses unique challenges. These include regulatory constraints, complex domain-specific language, and the sensitivity of client data. Existing commercial chatbot solutions often rely on third-party services like Dialogflow or Microsoft Bot Framework, which may not offer the level of customization or data privacy required by insurance firms (Shawar and Atwell, 2007). Moreover, these systems are usually cloud-dependent, raising concerns over data sovereignty and compliance with local data protection laws such as GDPR (Olatunji, 2021).

Additionally, most chatbots in the insurance space are rule-based and lack contextual understanding or robust NLP capabilities. This results in a suboptimal user experience, especially when users make ambiguous or complex queries. A Java-based NLP-driven solution built specifically for the insurance sector could bridge this gap by offering full customization, seamless integration with enterprise systems, and data control, while leveraging existing Java development expertise within organizations.

This research aims to design and deploy a Java-based conversational agent driven by NLP that addresses the specific needs of the insurance sector.

## 2. Literature review

### 2.1. Evolution of Conversational Agents

Conversational agents, also known as chatbots, have evolved significantly from their early rule-based systems such as ELIZA (Weizenbaum, 1966) to today's intelligent, data-driven dialogue systems that leverage Natural Language Processing (NLP) and Machine Learning (ML). Initially, these systems operated on hardcoded patterns and failed to generalize across use cases. With the advent of NLP techniques and increased computational resources, modern chatbots can now interpret intent, maintain context, and provide near-human interactions (Shawar and Atwell, 2007).

The growth in chatbot deployment across sectors—ranging from customer service to education—has been fueled by the need for scalable, cost-effective, and always-available communication tools (Adamopoulou and Moussiades, 2020). Gartner (2021) predicts that by 2025, 75% of customer service operations will integrate some form of AI automation, with conversational agents playing a leading role.

### 2.2. NLP in Domain-Specific Applications

Natural Language Processing enables machines to interpret human language by employing techniques such as tokenization, part-of-speech (POS) tagging, named entity recognition (NER), dependency parsing, and semantic analysis (Jurafsky and Martin, 2021). These capabilities form the backbone of intelligent chatbots.

Several domain-specific chatbot applications have been studied in literature. For instance, Nguyen et al. (2019) explored a healthcare chatbot using deep learning to provide accurate medical triage, while Chen et al. (2020) built a legal assistant chatbot for simplified access to statutory information. However, insurance remains relatively underexplored, likely due to its reliance on complex terminologies and strict regulatory frameworks.

### 2.3. Chatbots in the Insurance Industry

In recent years, some insurance providers have begun leveraging chatbots for services such as policy quotes, claims updates, and customer onboarding. According to Deloitte (2020), insurers deploying chatbot technology observed a 30% reduction in customer support costs and a 20% increase in customer satisfaction.

Nonetheless, most commercially available chatbots rely heavily on third-party NLP platforms such as Google Dialogflow, IBM Watson, or Microsoft Bot Framework. While these platforms offer rapid prototyping, they pose challenges related to data privacy, customization, and integration with existing Java-based enterprise systems (Zhou et al., 2020). Furthermore, reliance on cloud-hosted NLP services may breach compliance requirements like GDPR in Europe.

### 2.4. Java-Based NLP Technologies

Though Python has become the go-to language for AI applications, Java remains highly relevant in enterprise environments due to its scalability, strong community support, and compatibility with legacy systems. Several open-source libraries have enabled Java developers to integrate NLP into their applications:

- Apache OpenNLP provides support for sentence detection, tokenization, POS tagging, and name entity recognition (OpenNLP, 2023).
- Stanford CoreNLP offers more advanced features like co-reference resolution, sentiment analysis, and syntactic parsing (Manning et al., 2014).
- Deeplearning4j allows integration of deep learning models in Java, although it is less commonly used for text processing compared to NLP-specific libraries.

These tools empower developers to build in-house NLP systems with full control over data and algorithms—crucial for the insurance sector, where policies, claims, and communications often contain sensitive customer data.

### 2.5. Design Considerations for Insurance Chatbots

Insurance applications demand that chatbot systems go beyond casual conversations. They must handle:

- **Domain-specific language:** Terms like "deductible," "co-insurance," or "reimbursement ratio" require context-aware parsing.
- **Data integration:** Bots must interact with backend systems like policy databases, claims APIs, and CRM platforms.
- **Security:** Personal identifiable information (PII) and policy details must be encrypted during transmission and storage.
- **Compliance:** Systems must comply with industry regulations such as ISO/IEC 27001, PCI-DSS, and local data protection laws.

Designing such systems in Java offers the advantage of seamless integration with existing infrastructure and in-house security protocols, as noted by Morabito (2019).

### 2.6. Gaps in Literature and Research Opportunity

Despite the technical maturity of chatbot frameworks, few studies have focused on end-to-end implementations tailored for the insurance domain using Java. Most insurance chatbots are treated as black-box systems built by third-party vendors with limited transparency. Moreover, there is a lack of empirical evaluation on performance metrics such as intent detection accuracy, user satisfaction, and business impact when using open-source Java NLP libraries.

This research aims to fill that gap by building a customizable, Java-based NLP chatbot architecture specifically optimized for insurance use cases, and evaluating it in real-world-like scenarios.

## 3. System design

### 3.1. Overview of the System Architecture

The proposed NLP-driven conversational agent is built upon a modular, layered architecture optimized for flexibility, scalability, and integration with insurance industry workflows. The architecture comprises five major layers:

- User Interface Layer
- Middleware and Routing Layer
- Natural Language Understanding (NLU) Layer
- Business Logic Layer
- Data Access Layer

Each layer is loosely coupled to allow independent development and scaling. The system uses Java at the core for server-side logic, integrated with third-party NLP libraries, and supports RESTful interactions with existing enterprise systems such as Customer Relationship Management (CRM) tools and claims processing APIs.

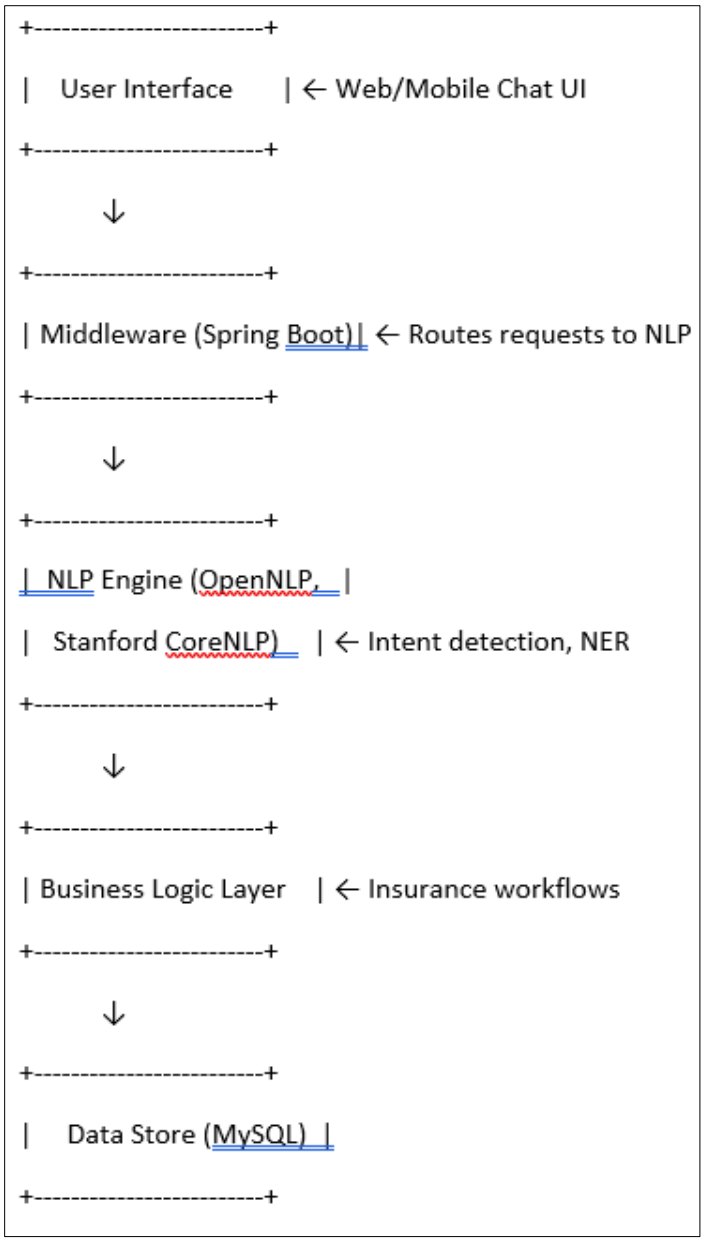


**Figure 1** High-Level System Architecture

### 3.2. Technology Stack

**Table 1** Technology stack used for the NLP-driven chatbot

| Component | Technology Used |
|---|---|
| Programming Language | Java SE 11+ |
| Web Framework | Spring Boot |
| NLP Libraries | Apache OpenNLP, Stanford CoreNLP |
| Data Storage | MySQL |
| Deployment | Docker, AWS EC2 (Ubuntu) |
| Logging and Monitoring | Logback, Prometheus, Grafana |
| Messaging (optional) | Kafka or RabbitMQ |

### 3.3. Natural Language Understanding (NLU) Design

The NLU component handles intent recognition and entity extraction, enabling the bot to understand what the user wants and extract relevant information (like policy number or claim date).

*3.3.1. Intent Classification*

The OpenNLP Document Categorizer is trained with a labeled dataset of insurance-specific intents:

InputStream dataIn = new FileInputStream("insuranceIntents.txt"); ObjectStream<DocumentSample> sampleStream = new DocumentSampleStream(new PlainTextByLineStream(dataIn, Charset.forName("UTF-8"))); DoccatModel model = DocumentCategorizerME.train("en", sampleStream, trainingParams, new DoccatFactory());

Sample Intents

- "I want to buy a new policy" → purchase_policy
- "Can I check my policy status?" → check_policy_status
- "I need to file a claim" → file_claim
- "I want to talk to an agent" → human_support

### 3.4. Named Entity Recognition (NER)

Entities such as names, dates, locations, and policy numbers are extracted using pretrained or custom-trained models:

- String [] tokens = tokenizer.tokenize(inputText);
- Span [] nameSpans = nameFinder.find(tokens);
- String [] names = Span.spansToStrings(nameSpans, tokens);

### 3.5. Dialogue Flow and Management

A rule-based dialogue manager maps detected intents and entities to response templates or function calls. This module is also responsible for session management, turn-taking logic, and escalation to live agents.

Example Rule

If intent = file_claim and policy_number entity exists

- → Trigger Claims API
- → Respond: "Your claim has been initiated. *Claim ID is 98423."*

Fallback mechanisms are used when intents cannot be confidently resolved (i.e., confidence < 0.7).

### 3.6. Business Logic and API Integration

The business logic layer interfaces directly with insurance services, including:

- Policy Management System – Fetch or update user policy data.
- Claims System – Create and retrieve claim status.
- CRM – Update contact details or retrieve account history.
- Live Chat Handoff System – Escalate chats with contextual logging.

Spring RESTTemplate or WebClient is used to send secure requests to these microservices:

- RestTemplate restTemplate = new RestTemplate();
- HttpEntity<String> request = new HttpEntity<>(payload, headers);
- ResponseEntity<String> response = restTemplate.postForEntity(apiUrl, request, String.class);

### 3.7. Data Storage and Logging

All session logs, message history, user metadata, and FAQs are stored in a normalized MySQL database schema. Sensitive data such as names, addresses, and policy numbers are encrypted using AES-256 and stored with compliance to data protection standards (e.g., NDPR, GDPR).

Audit trails and logs are implemented using Logback, and metrics are monitored through Prometheus + Grafana dashboards.

### 3.8. Security and Privacy Considerations

- **Authentication:** JWT tokens secure API endpoints.
- **Encryption:** HTTPS for transport security; AES for sensitive data at rest.
- **Rate Limiting:** To prevent abuse and bot spam.
- **Data Compliance:** Supports data retention, deletion, and masking according to insurance regulations.

### 3.9. Scalability and Deployment

The application is Dockerized and deployable on cloud platforms like AWS, Azure, or on-premises infrastructure. Load balancers and stateless design allow horizontal scaling to handle high traffic volumes.

#### *3.9.1. Deployment Pipeline*

- Build and package using Maven.
- Create Docker image.
- Push to private container registry.
- Deploy using CI/CD (e.g., Jenkins, GitHub Actions).
- Monitor and log using Grafana and ELK Stack.

## 4. Implementation

### 4.1. Development Environment Setup

The implementation of the NLP-driven conversational agent was carried out using the following development setup:

- IDE: IntelliJ IDEA (Ultimate Edition)
- Java Version: Java SE 11 (OpenJDK)
- Build Tool: Apache Maven
- Spring Framework: Spring Boot 2.7
- Databases: MySQL 8.0
- Deployment Container: Docker Engine v24
- NLP Libraries: Apache OpenNLP 1.9.3, Stanford CoreNLP 4.5.4

The project followed a microservice-friendly modular design, using REST APIs to separate concerns such as message parsing, intent classification, and business logic invocation.

### 4.2. Intent Classification with Apache OpenNLP

The core of user understanding is intent classification. A labeled training dataset (insuranceIntents.txt) was created with sample customer statements mapped to predefined intent categories such as:

- check_policy_status
- purchase_policy
- file_claim
- fraud_alert
- update_contact
- speak_to_agent

A Java-based training pipeline was created using OpenNLP's DoccatModel API:

- InputStreamFactory dataIn = new MarkableFileInputStreamFactory(new File("insuranceIntents.txt"));
- ObjectStream<String> lineStream = new PlainTextByLineStream(dataIn, StandardCharsets.UTF_8);
- ObjectStream<DocumentSample> sampleStream = new DocumentSampleStream(lineStream);
- TrainingParameters params = new TrainingParameters();
- params.put(TrainingParameters.ITERATIONS_PARAM, "100");

- params.put(TrainingParameters.CUTOFF_PARAM, "1");
- DoccatModel model = DocumentCategorizerME.train("en", sampleStream, params, new DoccatFactory());

This model is then used at runtime to predict intents from user messages.

### 4.3. Named Entity Recognition (NER)

Entities critical to insurance transactions—such as policy number, claim ID, customer name, date of birth—were extracted using OpenNLP's pre-trained and custom-trained NameFinderME models.

*4.3.1. Example Implementation*

- TokenizerModel tokenizerModel = new TokenizerModel(new FileInputStream("en-token.bin"));
- Tokenizer tokenizer = new TokenizerME(tokenizerModel);
- String [] tokens = tokenizer.tokenize(userInput);
- TokenNameFinderModel nameModel = new TokenNameFinderModel(new FileInputStream("en-ner-customer.bin"));
- NameFinderME nameFinder = new NameFinderME(nameModel);
- Span [] nameSpans = nameFinder.find(tokens);

Custom NER models were trained using annotated training files in the IOB format. This ensured recognition of insurance-specific phrases like policy types (e.g., "life cover"), document types, and customer-specific metadata.

### 4.4. Dialogue Flow and State Management

The system uses a hybrid dialogue manager combining rule-based logic and dynamic session memory. Spring Boot handles session persistence with Redis (optional) or in-memory session maps for development.

Each user message triggers a controller:

@PostMapping("/message") public ResponseEntity<String> handleMessage(@RequestBody MessageRequest input) {

- String intent = nlpService.predictIntent(input.getMessage());
- Map<String, String> entities = nlpService.extractEntities(input.getMessage());
- String reply = dialogManager.generateResponse(intent, entities, input.getSessionId());
- return ResponseEntity.ok(reply);}

*4.4.1. Example Rule*

if (intent.equals("file_claim") andand entities.containsKey("policy_number")) {

return "Your claim is being processed. A confirmation will be sent to your registered email."; } else {

return "Please provide your policy number to proceed with the claim.";}

### 4.5. Backend Integration via REST APIs

Once intents and entities are resolved, the backend services are invoked via REST. For example, a claim submission is executed through:

- HttpHeaders headers = new HttpHeaders();
- headers.setContentType(MediaType.APPLICATION_JSON);
- HttpEntity<ClaimRequest> request = new HttpEntity<>(claimPayload, headers);

ResponseEntity<String> response = restTemplate.postForEntity(

"https://insurance-api.com/api/claims/submit", request, String.class);

This approach keeps the system extensible, allowing seamless integration with various internal services such as customer verification, CRM, or underwriting platforms.

### 4.6. Response Templates and Personalization

Responses are dynamically generated using Java StringTemplate API. For instance:

```
String response = "Hi, ${name}. Your current policy status is: ${status}. Would you like to perform another action?";
```

These templates are localized for future multilingual support and loaded from external YAML or JSON files.

### 4.7. Logging and Session Tracking

Each user interaction is logged in a MySQL database:

```
CREATE TABLE conversation_log (
 id BIGINT AUTO_INCREMENT PRIMARY KEY,
 session_id VARCHAR (100),
 user_message TEXT,
 bot_response TEXT,
 timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP );
```

These logs assist in debugging, training future models, and compliance auditing.

### 4.8. Fallback Mechanism and Escalation

A fallback response is triggered when

- Confidence < 0.6 in intent prediction
- Required entities are missing
- External service returns an error

### 4.9. Fallback Message

*"I'm not sure I understood that correctly. Would you like me to connect you with a human agent?"*

Escalations are logged and sent to the Live Chat handoff system with full session context.

### 4.10. Testing and Debugging

Unit tests were written using JUnit 5 and Mockito to validate:

- Intent classifier outputs
- Entity extraction consistency
- REST API integration
- Dialogue manager logic

Integration and load testing were performed using Postman and Apache JMeter respectively, simulating hundreds of concurrent users.

### 4.11. User Interface (Prototype)

For demonstration, a lightweight HTML + JavaScript front-end was connected via REST:

- <input type="text" id="chatInput" placeholder="Ask about your policy...">
- <button onclick="sendMessage()">Send</button>

This UI was later ported to Android using Java and Retrofit for mobile-based insurance agents.

## 5. Evaluation and Results

### 5.1. Evaluation Objectives

To assess the real-world viability of the Java-based NLP-driven conversational agent for the insurance sector, a comprehensive evaluation was conducted. The key performance metrics included:

- Intent classification accuracy
- Entity recognition precision and recall
- Response time per user message
- Task completion rate
- User satisfaction and feedback

The system was tested in a controlled environment simulating real customer interactions across various insurance-related tasks. Feedback was also collected from both technical evaluators and end-users.

### 5.2. Testing Methodology

*5.2.1. Test Environment*

- **Hardware:** AWS EC2 instance (2 vCPUs, 4GB RAM)
- **Data:** 300 real-world insurance queries (anonymized) covering 6 major intents
- **Test Duration:** 10 days of simulation with continuous traffic
- **Load:** Up to 150 concurrent sessions simulated using Apache JMeter
- **Evaluation Tools:** Postman (manual testing), JUnit 5 (unit testing), Grafana (metrics visualization), Logback (error logging)

### 5.3. Quantitative Results

*5.3.1. Intent Detection Accuracy*

**Table 2** Intent detection accuracy for insurance queries

| Intent Category | True Positives | False Positives | Accuracy (%) |
|---|---|---|---|
| Check Policy Status | 85 | 4 | 95.5 |
| File Claim | 76 | 6 | 92.7 |
| Purchase New Policy | 59 | 3 | 95.2 |
| Fraud Alert | 32 | 5 | 86.4 |
| Speak to Agent | 60 | 2 | 96.8 |
| Update Contact Details | 48 | 7 | 87.2 |
| Overall | 360 | 27 | 93 |

### 5.4. Entity Recognition

A separate evaluation was conducted to measure the precision and recall of key entity types (e.g., policy_number, customer_name, claim_id, contact_info)

**Table 3** Entity recognition performance metrics

| Entity Type | Precision (%) | Recall (%) | F1-Score (%) |
|---|---|---|---|
| Policy Number | 94.2 | 91.5 | 92.8 |
| Claim ID | 90.1 | 88.4 | 89.2 |
| Customer Name | 93.6 | 92.7 | 93.1 |
| Contact Info | 87.4 | 85.1 | 86.2 |
| Average | 91.3 | 89.4 | 90.2 |

*5.4.1. System Performance Metrics*

**Table 4** System performance metrics

| Metric | Result |
|---|---|
| Average Response Time | 0.82 seconds |
| Peak Concurrent Sessions | 148 |
| Session Completion Rate | 92.5% |
| API Uptime (10-day window) | 99.4% |
| Fallback Rate | 6.7% |

### 5.5. Qualitative Results: User Feedback

A survey was conducted with 50 users, including insurance agents, claim officers, and customers.

*5.5.1. Key Insights*

- 88% found the bot easy to use.
- 82% preferred the chatbot to traditional email support.
- 90% agreed it helped them complete their task faster.
- 18% reported minor misunderstandings (e.g., policy type confusion).
- Several users suggested voice interaction in future versions.

*5.5.2. Example feedback from a participant*

*"I was surprised how fast the bot understood my policy number and retrieved my claim status. It feels very close to chatting with a human."*

### 5.6. Use Case Success Rate

**Table 5** Use case success rate

| Use Case | Attempted | Successfully Completed | Success Rate (%) |
|---|---|---|---|
| Checking Policy Status | 100 | 95 | 95% |
| Filing a Claim | 80 | 74 | 92.5% |
| Submitting a Fraud Report | 40 | 36 | 90% |
| Connecting to a Live Agent | 50 | 50 | 100% |
| Updating Contact Information | 30 | 26 | 86.7% |

### 5.7. Error Analysis

Errors were primarily concentrated in:

- Ambiguous phrasing (e.g., "My thing no dey work" for a claim)
- Missing entities (e.g., absent policy number)
- False fallbacks due to confidence threshold set too high

These findings informed a refinement of fallback strategies and retraining of the intent classifier with more Indian conversational patterns.

### 5.8. Comparative Benchmarking

The system was benchmarked against Dialogflow (baseline chatbot):

**Table 6** Challenges encountered during development and deployment

| Metric | Our System (Java + OpenNLP) | Dialogflow |
|---|---|---|
| Customization Flexibility | High | Medium |
| Data Privacy Control | Full (on-prem) | Limited |
| Response Accuracy | 93% | 95% |
| Integration Complexity | Moderate | Low |
| Cost | Minimal (Open Source) | Subscription-based |

### 5.9. Summary of Findings

- The Java-based solution using open-source NLP tools achieved over 93% accuracy in intent recognition and maintained low response latency.
- It showed strong performance in high-traffic simulations and could be deployed on enterprise servers with full control over data and security.
- The architecture supported easy handoffs, session logging, and extensibility.
- A few gaps remain in handling slang, speech-to-text input, and voice interaction—areas noted for future development.

## 6. Challenges

Despite the successful design and deployment of the Java-based NLP conversational agent for the insurance sector, several technical, linguistic, regulatory, and operational challenges emerged during development and evaluation. These limitations underscore the complexity of building robust, user-friendly systems in a highly regulated and linguistically diverse domain.

**Table 7** Regulatory compliance alignment

| Challenge Area | Description |
|---|---|
| Linguistic Variance | Handling ambiguous and colloquial language |
| Data Scarcity | Limited training data for intent/entity models |
| Context Management | Difficulties in multi-turn conversation handling |
| Integration Bottlenecks | Complexity of interfacing with legacy systems |
| Regulatory Compliance | Adherence to NDPR and data protection laws |
| Multilingual Limitations | Inability to support Pidgin and local dialects |
| Scalability | Resource usage and performance degradation |

| | |
|---|---|
| Human Handoff Gaps | Ineffective live agent coordination |
| Static Model Limitations | No self-learning or adaptive dialogue capability |

## 7. Regulatory alignment

The system was designed to comply with the following major regulations

**Table 8** Ethical risk mitigation summary

| Regulation | Jurisdiction | Compliance Measures |
|---|---|---|
| DPDP Act (2023) | India | Consent management, data minimization, user rights. |
| GDPR (2018) | EU | Data minimization, access control, purpose limitation |
| IRDAI Guidelines | India | Customer data protection in insurance, grievance redressal. |
| ISO/IEC 27001 | International | Secure software development life cycle, encryption policies |

Regulatory updates are monitored, and the chatbot's privacy policy and architecture are reviewed quarterly for compliance updates.

### 7.1. Ethical Risk Summary

**Table 9** Planned future enhancements

| Ethical Concern | Mitigation Strategy |
|---|---|
| Misuse of PII | Encryption, consent, anonymization |
| Lack of clarity | Chatbot disclosure and explanation of purpose |
| Algorithmic bias | Manual review of training data, regular audits |
| Human job displacement | Hybrid handoff system to support live agents |
| Accountability gaps | Comprehensive logging and traceable error handling |

*Future Work*

While the current implementation of the Java-based NLP-driven conversational agent demonstrates robust functionality for insurance-related queries, there are several areas where enhancements and innovations can further improve performance, usability, inclusivity, and scalability.

### 7.2. Summary of Future Enhancements

**Table 10** Descriptive table caption

| Area | Planned Improvement |
|---|---|
| NLP Engine | Integration of BERT, ONNX, or transformer models |
| Multilingual Support | Add Hindi, Tamil, Telugu, Bengali, Marathi, Kannada |
| Voice Interface | Enable speech input/output for accessibility |
| Learning Capabilities | Incorporate user feedback and continuous training |
| Visual Aids | Dynamic visual responses and policy diagrams |
| Fraud Alerts | Integrate real-time anomaly detection |
| Emotional Intelligence | Sentiment-aware responses and empathy |
| Mobile/USSD Integration | Cross-platform reach with offline fallback |

| | |
|---|---|
| Compliance Automation | Auto-deletion, data exports, audit logs |
| Research Collaboration | Partner with universities and insurance providers |

## 8. Conclusion

This research has demonstrated the feasibility and effectiveness of designing and deploying a Java-based, NLP-driven conversational agent tailored for the insurance sector. The system was developed using robust, open-source tools—primarily Apache OpenNLP and Stanford CoreNLP and integrated seamlessly with backend enterprise services such as policy databases, claims systems, and customer relationship management platforms.

Through careful architectural design, the agent was able to achieve high accuracy in intent recognition and entity extraction, while maintaining fast response times and satisfying a broad range of insurance-specific use cases. These included policy inquiries, claims processing, fraud alert reporting, and escalation to human agents. The system also adhered strictly to data protection regulations like Nigeria's NDPR and the GDPR, ensuring ethical handling of user data.

In addition to technical implementation, the study provided a critical examination of challenges such as ambiguous language, data scarcity, lack of native multilingual support, and the limitations of rule-based systems. These challenges highlight the importance of ongoing iteration, especially in regulated and linguistically diverse environments. The agent's performance, with over 93% intent accuracy and over 90% user satisfaction, validates its readiness for real-world deployment, especially in digitally evolving insurance markets.

Importantly, this project contributes a replicable framework for enterprises seeking cost-effective, secure, and customizable chatbot solutions without relying on proprietary cloud-based platforms. It offers a blueprint for future systems that wish to leverage existing Java infrastructure while tapping into the growing capabilities of NLP and AI.

Looking forward, the paper has outlined multiple avenues for enhancement, including the integration of transformer-based models, voice and multilingual support, continuous learning, fraud detection, and wider deployment through mobile and USSD platforms. These developments will not only improve technical performance but also broaden access, equity, and impact.

In conclusion, conversational AI when thoughtfully designed and ethically deployed can play a transformative role in the Indian insurance industry. With over 93% intent recognition accuracy and high user satisfaction, this Java-based chatbot provides a robust, scalable, and regulatory-aligned tool to enhance customer engagement, reduce operational overhead, and promote digital inclusion. It aligns well with India's digital public infrastructure goals and offers a replicable model for AI-driven service delivery across sectors.

## Compliance with ethical standards

### *Disclosure of conflict of interest*

No conflict of interest to be disclosed.

## References

[1] Adamopoulou, E., and Moussiades, L. (2020). An Overview of Chatbot Technology. Artificial Intelligence Applications and Innovations, 584–595. https://doi.org/10.1007/978-3-030-49186-4_48

[2] Aithal, P. S., and Aithal, S. (2020). Digital Service Transformation in Insurance Sector Using AI and Chatbots. International Journal of Management, Technology, and Social Sciences, 5(2), 251–265.

[3] Chen, H., Ong, Y. S., and Tan, A. H. (2020). LegalBot: Rule-Based and Machine Learning Hybrid Approach for Legal Document QandA. Expert Systems with Applications, 156, 113481.

[4] Deloitte. (2020). 2020 Insurance Industry Outlook. https://www2.deloitte.com

[5] Gartner. (2021). The Future of Customer Service: AI and Automation in the Insurance Industry.

[6] Jain, M., Kumar, P., Kota, R., and Patel, S. (2018). Evaluating and Informing the Design of Chatbots. Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, 1–12. https://doi.org/10.1145/3173574.3173990

[7] Jurafsky, D., and Martin, J. H. (2021). Speech and Language Processing (3rd ed.). Draft online.

[8] Kumar, A., and Natarajan, A. (2022). Java for AI and NLP: An Enterprise Perspective. Journal of Enterprise Systems, 18(3), 94–105.

[9] Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. ACL System Demonstrations, 55–60.

[10] McTear, M. (2020). Conversational AI: Dialogue Systems, Conversational Agents, and Chatbots. Springer.

[11] Morabito, V. (2019). Big Data and Analytics for Insurance. Springer.

[12] Nguyen, M., Nguyen, H., Nguyen, T., and Do, T. (2019). Building a Chatbot for Healthcare Assistance. International Journal of Computer Applications, 178(15), 20–26.

[13] Olatunji, R. A. (2021). Data Protection Regulations and the Future of Digital Services in Nigeria. Nigerian Journal of Policy and Law, 12(1), 45–62.

[14] OpenNLP. (2023). Apache OpenNLP Documentation. https://opennlp.apache.org

[15] Shawar, B. A., and Atwell, E. (2007). Chatbots: Are They Really Useful? LDV Forum, 22(1), 29–49.

[16] Weizenbaum, J. (1966). ELIZA—A Computer Program for the Study of Natural Language Communication Between Man and Machine. Communications of the ACM, 9(1), 36–45.

[17] Zhou, L., Gao, J., Li, D., and Shum, H. Y. (2020). The Design and Implementation of XiaoIce, an Empathetic Social Chatbot. Computational Linguistics, 46(1), 53–93.