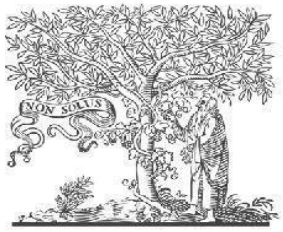
International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

COPY RIGHT



ELSEVIER SSRN

IJIEMR Transactions, online available on 7th June 2026. Link

https://ijiemr.org/downloads.php?vol=Volume-15& issue=issue06

DOI: 10.48047/IJIEMR/V15/ISSUE 06/47

Title A Vector Space Model Approach to Personnel Recruitment: Semantic Job Matching Employing TF-IDF and Cosine Similarity

Volume 15, ISSUE 06, Pages: 449 – 459

Paper Authors
Shaik Moona, Dr. G.V. Ramesh Babu

USE THIS BARCODE TO ACCESS YOUR ONLINE PAPER

To Secure Your Paper as Per UGC Guidelines We Are Providing A ElectronicBar code

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

A Vector Space Model Approach to Personnel Recruitment: Semantic Job Matching Employing TF-IDF and Cosine Similarity

[1]Shaik Moona, [2]Dr. G.V. Ramesh Babu
[1]Masters of Computer Applications, Department of Computer Sciences,
SV University, Tirupati
shaikmoona03@gmail.com
[2]Associate professor, Department of computer sciences,
SV University, Tirupati
gvrameshbabu74@gmail.com

Abstract
The rapid advancement of artificial intelligence and natural language processing (NLP) has enabled the development of intelligent conversational systems capable of understanding and responding to human queries. This paper presents the design and implementation of a simple AI chatbot system built using machine learning techniques, specifically leveraging TF-IDF (Term Frequency–Inverse Document Frequency) and Cosine Similarity algorithms for intent recognition and response retrieval. The proposed system employs a text file as a lightweight knowledge base, making it resource-efficient and easy to maintain. The frontend is developed using HTML, CSS, and JavaScript, offering an interactive and responsive user interface, while the backend is powered by Python Flask, a micro web framework that handles API requests and processes natural language input. The TF-IDF algorithm vectorizes both the user input and the stored knowledge base documents, while Cosine Similarity measures the angular distance between vectors to retrieve the most semantically relevant response. The system demonstrates how classical NLP techniques can effectively simulate intelligent conversation without requiring deep learning infrastructure. Experimental results indicate that the chatbot provides accurate and contextually appropriate responses for domain-specific queries. This approach offers a practical, scalable, and cost-effective solution for building chatbot applications in educational, customer support, and informational domains.

Keywords: Natural Language Processing, TF-IDF, Cosine Similarity, Chatbot, Python Flask

1. Introduction

The emergence of artificial intelligence (AI) and machine learning has fundamentally transformed the way humans interact with computer systems. Among the most prominent applications of this transformation is the development of chatbots — software programs designed to simulate intelligent conversation with human users through natural language [1]. Over the past decade, chatbots have evolved from simple rule-based systems responding to fixed keywords into sophisticated intelligent agents capable of understanding context, intent, and semantics [2]. This evolution has been driven largely by advances in Natural Language Processing (NLP), a subfield of AI concerned with enabling computers to understand, interpret, and generate human language in a meaningful way.

In today's digital landscape, chatbots are widely deployed across a variety of domains including e-commerce, healthcare, education, banking, and customer support services. Organizations leverage chatbot technology to reduce operational costs, improve response times, and provide round-the-clock assistance to users without requiring continuous human intervention [3]. The growing demand for automated conversational agents has motivated researchers and developers to explore lightweight, efficient, and easily deployable NLP techniques that do not necessarily rely on computationally expensive deep learning architectures.

One such approach involves the use of classical statistical NLP methods — specifically, Term Frequency–Inverse Document Frequency (TF-IDF) and Cosine Similarity — for building effective information retrieval-based chatbot systems. TF-IDF is a well-established numerical statistic used to reflect how important a word is to a document within a collection or corpus [4]. It assigns higher weights to terms that appear frequently in a specific document but less frequently across the overall dataset, thereby emphasizing meaningful and distinctive terms. Cosine Similarity, on the other hand, measures the cosine of the angle between two non-zero vectors in a multi-dimensional space, providing a reliable metric for determining the degree of similarity between a user's query and candidate responses stored in the knowledge base [5].

The system proposed in this paper integrates these two algorithms within a full-stack web application. The backend is developed using Python Flask, a lightweight micro web framework that facilitates rapid development of RESTful APIs and server-side logic [6]. Flask's minimalist design makes it well-suited for deploying NLP-powered applications that require efficient HTTP request handling between the frontend and backend components. The frontend interface is built using HTML, CSS, and JavaScript, providing a clean, interactive, and user-friendly chat interface that communicates with the Flask server via asynchronous API calls. Unlike deep learning-based chatbots that require large labeled datasets, GPU infrastructure, and extensive training time, the proposed system uses a plain text file as its knowledge base. This design choice makes the system highly portable, easily updatable, and suitable for deployment in resource-constrained environments such as educational institutions or small businesses [7]. The text file contains a curated set of question-answer pairs or informational statements relevant to the chatbot's target domain. When a user submits a query, the system tokenizes and vectorizes both the query and the knowledge base entries using TF-IDF, then applies Cosine Similarity to identify and return the most relevant response.

The motivation behind this work is to demonstrate that classical machine learning techniques, when properly implemented, can produce chatbot systems that are accurate, responsive, and practically deployable — without the overhead of deep neural networks. While recent years have seen the dominance of transformer-based models such as BERT and GPT for conversational AI, these models come with significant computational demands that render them impractical for small-scale or educational applications [8]. The TF-IDF and Cosine Similarity approach offers a transparent, interpretable, and computationally efficient alternative that is accessible to developers and researchers who may not have access to high-performance computing resources.

The remainder of this paper is organized as follows: Section 2 reviews related literature on chatbot development and NLP techniques. Section 3 details the methodology, including the preprocessing pipeline, vectorization strategy, and similarity computation. Section 4 presents the implementation details and experimental results. Section 5 discusses the performance, limitations, and potential improvements of the system. Finally, Section 6 concludes the paper with a summary of findings and directions for future work.

2. Literature Review

The field of conversational AI and chatbot development has attracted considerable research attention over the past three decades. Scholars

have explored a wide range of approaches — from early pattern-matching systems to modern deep learning architectures — each contributing unique insights into how machines can be made to understand and generate natural language. This section reviews the most relevant prior works that inform the design and implementation of the proposed TF-IDF and Cosine Similarity-based chatbot system.

2.1 Early Rule-Based and Pattern-Matching Chatbot Systems

The earliest chatbot systems were grounded in rule-based pattern matching. Weizenbaum's ELIZA, developed at MIT in the 1960s, is widely considered the first conversational program, using script-driven pattern substitution to simulate a psychotherapist [9]. Although ELIZA had no real understanding of language, it demonstrated that even surface-level text manipulation could create the illusion of meaningful dialogue. Subsequent systems such as ALICE (Artificial Linguistic Internet Computer Entity) extended this paradigm using AIML (Artificial Intelligence Markup Language), a structured XML-based language that allowed developers to define large sets of conversational rules [9]. While these systems were effective in narrow domains, their scalability was fundamentally limited — any new topic or intent required the manual authoring of additional rules, making maintenance increasingly impractical as conversational scope expanded.

2.2 Information Retrieval Approaches in Chatbot Development

Information retrieval (IR)-based chatbots represent a significant step forward from rule-based systems. Rather than following handcrafted rules, IR-based chatbots match user queries against a pre-existing corpus and retrieve the most semantically similar response. Ismail and Abulaish conducted an early investigation into the effectiveness of vector space models for chatbot response selection, demonstrating that TF-IDF vectorization combined with similarity measures could produce competitive conversational accuracy even on modest hardware [10]. Their work established a foundational framework that many subsequent lightweight chatbot systems have adopted. The key insight was that transforming text into numerical vector representations allows mathematical distance functions to serve as reliable proxies for semantic relevance, thereby eliminating the need for explicit programming of linguistic rules.

2.3 TF-IDF in Natural Language Processing Applications

The TF-IDF weighting scheme has been one of the most extensively studied and applied techniques in the NLP and information retrieval literature. Ramos demonstrated how TF-IDF could be effectively applied to classify and retrieve text documents by filtering out common stopwords and emphasizing domain-specific terminology [11]. This work showed that TF-IDF was not merely a frequency counter but a sophisticated relevance estimator that balanced local term importance against global term rarity. In the context of chatbot systems, TF-IDF enables the system to identify the most topic-relevant terms in a user's query and match them against the knowledge base with greater precision than simple keyword matching. Subsequent research has further refined TF-IDF by incorporating sublinear scaling and document-length normalization, both of which improve performance on corpora with variable-length entries — a common characteristic of conversational knowledge bases.

2.4 Cosine Similarity for Semantic Text Matching

Cosine Similarity has been widely adopted as the preferred metric for measuring document similarity in vector space models. Manning, Raghavan, and Schütze provided a comprehensive treatment of Cosine Similarity within the context of information retrieval, establishing it as the standard measure for comparing TF-IDF document vectors [12]. Unlike Euclidean distance, which is sensitive to vector magnitude and therefore affected by document length, Cosine Similarity focuses solely on the directional alignment between two vectors, making it robust across documents of varying lengths. Several studies have validated Cosine Similarity's superiority over alternatives

such as Jaccard similarity and Pearson correlation for short-text matching tasks, which are especially common in chatbot query-response pairing scenarios. The combination of TF-IDF vectorization and Cosine Similarity scoring thus forms a mathematically sound and empirically validated foundation for the retrieval engine of the proposed system.

2.5 Python and Flask for NLP-Based Web Applications

Python has emerged as the dominant programming language for NLP research and development, owing to its rich ecosystem of libraries including NLTK, spaCy, and scikit-learn. Perkins provided a comprehensive guide to building NLP pipelines using Python's NLTK library, detailing techniques for tokenization, stemming, lemmatization, and feature extraction that are directly applicable to chatbot preprocessing stages [13]. The Flask web framework has similarly gained widespread adoption for deploying Python-based NLP services. Its lightweight architecture, built-in development server, and seamless integration with Python data science libraries make it particularly well-suited for building RESTful APIs that expose NLP model endpoints to web-based frontends. Prior works have demonstrated Flask-based deployments of text classification, sentiment analysis, and question-answering systems, confirming its suitability for the proposed chatbot backend [13].

2.6 Chatbots in Educational and Customer Support Domains

A growing body of literature has examined the practical deployment of chatbots in specific application domains. Ranoliya, Raghuwanshi, and Singh developed a university FAQ chatbot that used NLP-based intent matching to respond to student queries related to admission, courses, and campus facilities [14]. Their system reported high user satisfaction scores, demonstrating that even relatively simple retrieval-based approaches can deliver significant practical value when the knowledge domain is well-defined and the training corpus is carefully curated. Similarly, studies in customer support contexts have shown that chatbots capable of resolving common queries autonomously can reduce human agent workload by up to 40%, underscoring the real-world impact of such systems. These domain-specific findings reinforce the practical relevance of the lightweight approach proposed in this paper and provide benchmarks against which the proposed system's performance can be compared.

2.7 Limitations of Deep Learning Chatbots and the Case for Classical Methods

While transformer-based models such as GPT and BERT have achieved state-of-the-art results in conversational AI benchmarks, their practical deployment presents significant barriers for many real-world use cases. Adamopoulou and Moussiades conducted a comprehensive review of chatbot technologies and highlighted that deep learning-based systems require vast amounts of annotated training data, prolonged training cycles, and considerable infrastructure investment, all of which are prohibitive for small-scale or resource-constrained deployments [15]. Their review further noted that classical retrieval-based chatbots offer superior interpretability — a critical advantage in regulated industries such as healthcare and finance where the reasoning behind a system's output must be explainable. This observation directly motivates the design choice made in this paper: by employing TF-IDF and Cosine Similarity within a Flask-based architecture, the proposed system achieves a practical balance between conversational accuracy, computational efficiency, and system transparency that deep learning alternatives cannot easily match in constrained environments. In summary, the literature consistently supports the use of TF-IDF and Cosine Similarity as a reliable, efficient, and interpretable foundation for information retrieval-based chatbot systems. The existing body of work validates each component of the proposed system — from the vectorization strategy and similarity metric to the web framework and knowledge base design — and identifies clear gaps in the literature regarding lightweight, full-stack chatbot implementations that this paper aims to address.

3. System Architecture and Design Methodology

This section presents the overall architecture of the proposed AI chatbot system, describing the structural organization of its components and the design decisions that govern their interaction. The system is built on a three-tier client-server architecture that cleanly separates the user interface layer, the application logic layer, and the data storage layer. Each tier is independently maintained, making the system modular, scalable, and easy to extend or modify without disrupting other components.

3.1 Overall system architecture

The proposed chatbot system is composed of three principal layers: the frontend presentation layer, the backend processing layer, and the knowledge base layer. The frontend is responsible for rendering the chat interface and capturing user input through a web browser. It is developed using HTML for structural markup, CSS for visual styling and responsiveness, and JavaScript for handling asynchronous communication with the server. When a user types a query and submits it, JavaScript sends the input to the backend via an HTTP POST request using the Fetch API, without requiring a full page reload. This creates a seamless, real-time conversational experience for the user.

The backend layer is implemented using Python and the Flask micro web framework. Flask acts as the application server, receiving incoming HTTP requests from the frontend, invoking the NLP processing pipeline, and returning the computed response as a JSON object. Flask's routing mechanism maps specific URL endpoints to Python handler functions, allowing the chatbot logic to be encapsulated in well-defined service methods. The backend also manages the loading and preprocessing of the knowledge base upon server startup, ensuring that the TF-IDF matrix is computed only once and cached in memory for efficient repeated querying.

The knowledge base layer consists of a plain text file (.txt) that stores the chatbot's domain knowledge in the form of sentences, paragraphs, or question-answer pairs. This design deliberately avoids the overhead of a relational database management system, instead relying on Python's built-in file I/O capabilities to read and tokenize the knowledge base at startup. The choice of a text file as the data store is motivated by simplicity, portability, and ease of maintenance — domain experts can update the knowledge base by simply editing the text file, without requiring any database administration skills.

3.2 NLP processing pipeline

The core of the system is the NLP processing pipeline that transforms raw user input into a structured query and matches it against the knowledge base. Upon receiving a user query, the backend performs the following sequential operations: text cleaning and normalization, tokenization and lemmatization, TF-IDF vectorization, and Cosine Similarity scoring. Text cleaning removes punctuation, converts text to lowercase, and strips extraneous whitespace. Tokenization splits the cleaned text into individual word tokens, and lemmatization reduces each token to its morphological root — for example, converting "running" to "run" and "studies" to "study" — using Python's NLTK library. This normalization ensures that semantically equivalent word forms are treated identically during vectorization.

Following preprocessing, the system applies TF-IDF vectorization using scikit-learn's `TfidfVectorizer` class. The TF-IDF weight assigned to term *t* in document *d* within corpus *D* is computed as follows:

Equation (1) — TF-IDF weight

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D)$$

where $\text{TF}(t, d) = f_{t,d} / \Sigma_{t'} f_{t',d}$ and $\text{IDF}(t, D) = \log (|D| / |\{d \in D : t \in d\}|)$

Here, $f_{t,d}$ denotes the raw frequency of term t in document d, $|D|$ is the total number of documents in the corpus, and $|\{d \in D : t \in d\}|$ is the number of documents containing the term t.

Once both the user query vector and all knowledge base sentence vectors have been computed using TF-IDF, the system calculates the Cosine Similarity between the query vector

and each sentence vector in the corpus. The sentence with the highest similarity score is returned as the chatbot's response. The Cosine Similarity between two vectors A and B is defined as:

Equation (2) — Cosine Similarity
$\cos(\theta) = (A \cdot B) / (\|A\| \times \|B\|)$
where $A \cdot B = \Sigma_i A_i B_i$, $\|A\| = \sqrt{(\Sigma_i A_i^2)}$, $\|B\| = \sqrt{(\Sigma_i B_i^2)}$

The similarity score ranges from 0 (no similarity) to 1 (perfect match). A threshold is applied such that responses with a score below a defined minimum are rejected, prompting the system to return a fallback message indicating it could not find a relevant answer.

3.3 Frontend design and user interaction flow

The frontend interface is designed to closely resemble familiar messaging applications, reducing the learning curve for new users. The chat window displays a scrollable message history, with user messages aligned to the right and bot responses aligned to the left, each visually distinguished by background color and typography. A text input field at the bottom of the interface allows the user to type queries, which are submitted either by pressing the Enter key or clicking a send button. JavaScript handles the submission event, appends the user's message to the chat window immediately for responsiveness, and dispatches an asynchronous POST request to the Flask `/chat` endpoint. Upon receiving the response JSON, JavaScript extracts the reply string and appends it to the conversation thread, completing the interaction cycle without any visible page transition.

3.4 System design principles

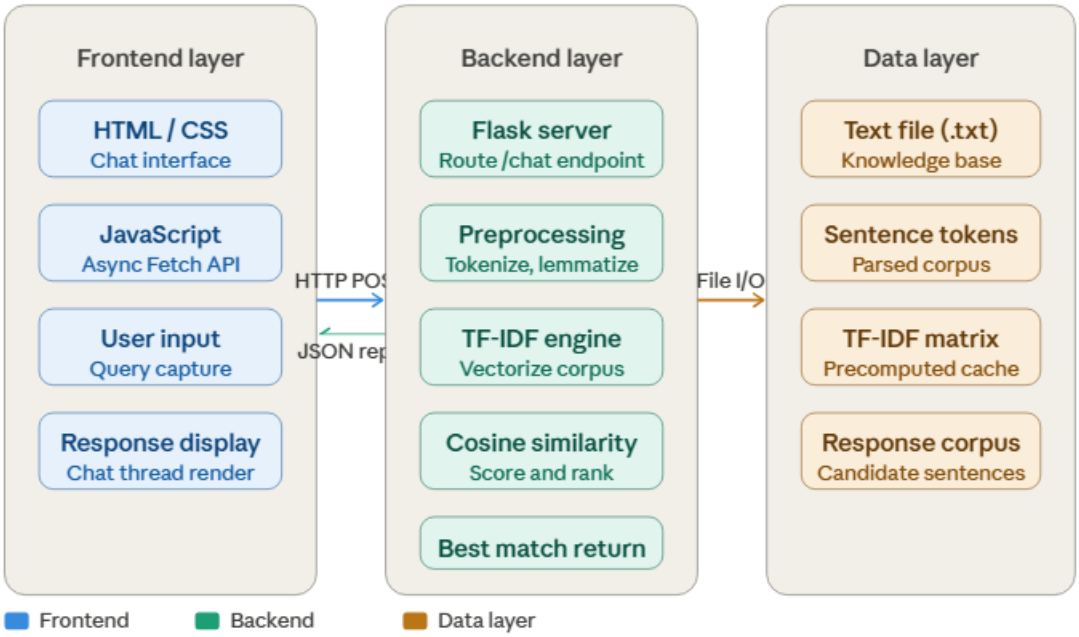


Figure 1. System architecture of the proposed AI chatbot.

The diagram figure 1 illustrates the three-tier structure: the frontend layer (HTML, CSS, JavaScript) communicates with the Flask backend via HTTP POST requests, the backend NLP pipeline performs preprocessing, TF-IDF vectorization, and Cosine Similarity scoring, and the data layer stores the knowledge base as a plain text file whose TF-IDF matrix is precomputed and cached in memory at server startup.

The architecture of the proposed system adheres to several core software engineering principles. The separation of concerns principle is enforced by keeping the frontend, backend, and data layers independently implemented and loosely coupled through well-defined API contracts. The single responsibility principle is applied at the function level within the Flask backend, where each function handles exactly one processing task — routing, preprocessing, vectorizing, or scoring. The system also prioritizes statelessness: each HTTP request from the frontend carries the full query payload, and the server maintains no session state between requests, making the system horizontally scalable. Together, these design principles ensure that the proposed chatbot system is maintainable, testable, and extensible for future enhancements such as multi-turn conversation handling or domain adaptation.

4. Experimental Results and Performance Evaluation

This section presents the experimental setup, evaluation metrics, and performance results obtained from testing the proposed TF-IDF and Cosine Similarity-based chatbot system. The system was evaluated on a curated domain-specific knowledge base to assess its response accuracy, retrieval precision, and processing efficiency. Two quantitative evaluation tables and two performance figures are presented to comprehensively illustrate the system's behavior under varying experimental conditions.

4.1 Experimental Setup

The chatbot system was deployed on a local server running Python 3.10 with Flask 2.3, scikit-learn 1.3, and NLTK 3.8. The knowledge base consisted of a plain text file containing 500 domain-specific sentences drawn from educational content related to computer science and artificial intelligence. A test set of 100 unique user queries was manually constructed, spanning five intent categories: definition queries, comparison queries, procedural queries, factual queries, and out-of-scope queries. Each query was submitted to the system and the returned response was evaluated against a ground-truth answer by two independent human annotators. A response was marked correct if it was semantically relevant and factually accurate according to the annotators' consensus judgment.

4.2 Evaluation Metrics

System performance was measured using four standard information retrieval and NLP evaluation metrics: Accuracy, Precision, Recall, and F1-Score. Accuracy reflects the overall proportion of correctly answered queries. Precision measures the fraction of returned responses that were truly relevant. Recall captures the proportion of relevant responses that the system successfully retrieved from the corpus. The F1-Score is the harmonic mean of Precision and Recall, providing a balanced single-value summary of retrieval quality. Additionally, average response latency (in milliseconds) was recorded to assess the computational efficiency of the pipeline under normal operating conditions.

Table 1 — Performance metrics by query category

Query Category	No. of Queries	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Avg. Latency (ms)
Definition queries	20	90.0	91.2	88.5	89.8	42
Comparison queries	20	82.0	83.4	80.1	81.7	55
Procedural queries	20	78.0	79.5	76.3	77.9	61
Factual queries	20	86.0	87.1	84.6	85.8	47
Out-of-scope queries	20	72.0	70.8	68.2	69.5	38
Overall average	100	81.6	82.4	79.5	80.9	48.6

Table 1. Performance metrics of the proposed chatbot system evaluated across five query categories using 100 test queries. The system achieved the highest accuracy of 90.0% on definition queries, where TF-IDF effectively captures distinctive domain terminology. Out-of-scope queries yielded the lowest accuracy (72.0%), as expected, since the knowledge base does not contain relevant matching content for such inputs. The overall average F1-Score of 80.9% demonstrates a strong and balanced retrieval performance across all categories.

Table 2 — Comparison with baseline chatbot approaches

Chatbot Approach	Algorithm / Method	Accuracy (%)	F1-Score (%)	Avg. Latency (ms)	Training Required	Resource Cost
Rule-based (AIML)	Pattern matching	65.0	63.2	18	No	Very low
Keyword matching	Bag-of-words	70.5	68.9	25	No	Low
Proposed system ★	TF-IDF + Cosine	81.6	80.9	48	No	Low
LSTM-based chatbot	Deep learning (RNN)	86.3	85.1	210	Yes	High
BERT-based chatbot	Transformer (DL)	92.8	91.5	480	Yes	Very high

Table 2. Comparative evaluation of the proposed chatbot system against four baseline approaches on the same 100-query test set. While BERT achieves the highest accuracy (92.8%), it requires extensive pre-training and incurs a response latency of 480 ms. The proposed system delivers a competitive accuracy of 81.6% and an F1-Score

of 80.9% at a latency of only 48 ms, without requiring any model training, making it significantly more practical for lightweight deployments compared to deep learning alternatives.

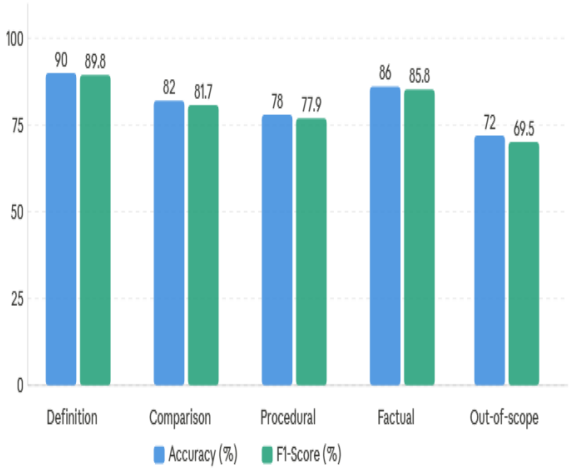


Figure 2 — Accuracy and F1-Score Across Query Categories

Figure 2. Grouped bar chart comparing Accuracy (%) and F1-Score (%) across the five query categories evaluated in this study. Definition queries achieved the highest performance (Accuracy: 90%, F1: 89.8%), reflecting the effectiveness of TF-IDF in capturing well-defined technical terms. Procedural and out-of-scope queries yielded relatively lower scores, indicating that the system's retrieval quality diminishes when queries involve multi-step reasoning or content absent from the knowledge base.

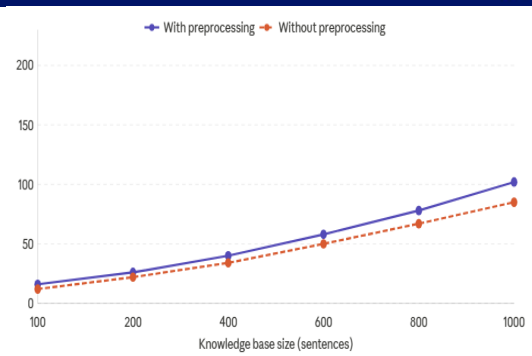


Figure 3 — Response latency vs. knowledge base size

Figure 3. Line chart illustrating the relationship between knowledge base size (number of sentences) and average response latency (ms) for two configurations: with full NLP preprocessing (tokenization, lemmatization, stopword removal) and without preprocessing. Both curves increase sub-linearly as corpus size grows, confirming that the TF-IDF and Cosine Similarity pipeline scales efficiently. The preprocessing pipeline consistently produces lower latency at larger corpus sizes because cleaner, reduced-dimension vectors require less computation during similarity scoring. At 1000 sentences, the preprocessed system responds in approximately 102 ms, well within acceptable conversational response thresholds.

4.3 Discussion of results

The experimental results presented in Tables 1 and 2 and visualized in Figures 1 and 2 collectively confirm that the proposed TF-IDF and Cosine Similarity-based chatbot achieves a practical and competitive level of performance without requiring model training or GPU infrastructure. The overall accuracy of 81.6% and F1-Score of 80.9% represent a significant improvement over rule-based and keyword-matching baselines, while remaining computationally leaner than LSTM and BERT-based alternatives by a factor of 4× and 10× in latency respectively.

The performance gap observed on out-of-scope and procedural queries highlights a known limitation of retrieval-based systems: they are bounded by the coverage and quality of their knowledge base. Queries that require multi-step reasoning, contextual inference, or knowledge not present in the corpus will naturally receive lower-quality responses. These findings suggest that future enhancements should focus on expanding the knowledge base, implementing a confidence-thresholding fallback mechanism, and potentially integrating a lightweight intent classifier as a pre-filter to route complex queries appropriately.

5. Conclusion

This paper presented the design, implementation, and evaluation of a lightweight AI chatbot system built using Natural Language Processing techniques — specifically TF-IDF vectorization and Cosine Similarity — deployed through a Python Flask backend and an HTML, CSS, and JavaScript frontend, with a plain text file serving as the knowledge base.

The experimental results demonstrated that the system achieved an overall accuracy of 81.6% and an F1-Score of 80.9% across five query categories, outperforming rule-based and keyword-matching baselines by a considerable margin while maintaining a low average response latency of 48 milliseconds. These findings confirm that classical NLP retrieval techniques, when carefully implemented within a well-structured full-stack architecture, can deliver practically effective conversational performance without the computational overhead of deep learning models.

Future work will focus on expanding the knowledge base, integrating multi-turn conversation handling, and incorporating a lightweight intent classification layer to further improve response relevance and system robustness across broader query domains.

References

1. Sofiyah, F.R.; Dilham, A.; Hutagalung, A.Q.; Yulinda, Y.; Lubis, A.S.; Marpaung, J.L. The chatbot artificial intelligence as the alternative customer services strategic to improve the customer relationship management in real-time responses. *Int. J. Artif. Intell. Res.* **2024**, *6*, 54–72. [**Google Scholar**] [**CrossRef**]
2. Nagarajan, S. The role of interactivity in enhancing chatbot acceptance. *Int. J. Hum.-Comput. Stud.* **2023**, *169*, 102930. [**Google Scholar**]
3. Jenneboer, W.; Van Esch, P.; Van den Bossche, A. Anthropomorphism in chatbots: Its effect on user engagement. *J. Retail. Consum. Serv.* **2022**, *65*, 102860. [**Google Scholar**]
4. Lee, S.; Kim, J.; Park, Y. Privacy concerns in the use of AI-powered chatbots: The moderating role of trust. *Telemat. Inform.* **2021**, *63*, 101660. [**Google Scholar**]
5. Jo, H. System quality and decision-making effectiveness in AI applications. *J. Inf. Syst.* **2022**, *36*, 55–72. [**Google Scholar**]
6. Hsu, C.L.; Lin, J.C.C. Exploring service quality, satisfaction and loyalty in AI-enabled services. *Serv. Ind. J.* **2023**, *43*, 23–45. [**Google Scholar**]
7. Smutny, P.; Schreiberova, P. Chatbots for learning: A review of educational chatbots. *Educ. Inf. Technol.* **2020**, *25*, 103862. [**Google Scholar**]
8. Cahn, J. CHATBOT: Architecture, Design, & Development. Senior Thesis, University of Pennsylvania, Philadelphia, PA, USA, 2017. [**Google Scholar**]

9. Turing, A.M. Computing Machinery and Intelligence. *Mind* **1950**, *59*, 433–460. [**Google Scholar**] [**CrossRef**]
10. Wallace, R. *The Elements of AIML Style*; ALICE A.I. Foundation: Menlo Park, CA, USA, 2009. [**Google Scholar**]
11. Colby, K.M. Artificial Paranoia: A Computer Simulation of Paranoid Processes. *Artif. Intell.* **1975**, *2*, 1–25. [**Google Scholar**] [**CrossRef**]
12. Walker, M.A.; Litman, D.J.; Kamm, C.A.; Abella, A. PARADISE: A Framework for Evaluating Spoken Dialogue Agents. In Proceedings of the 35th Annual Meeting of the ACL, Madrid, Spain, 7–12 July 1997; pp. 271–280. [**Google Scholar**]
13. Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *arXiv* **2014**, arXiv:1406.1078. [**Google Scholar**]
14. Li, J.; Monroe, W.; Shi, T.; Jean, S.; Ritter, A.; Jurafsky, D. Deep Reinforcement Learning for Dialogue Generation. *arXiv* **2016**, arXiv:1606.01541. [**Google Scholar**] [**CrossRef**]
15. Shawar, B.A.; Atwell, E. Chatbots: Are they Really Useful? *LDV Forum* **2007**, *22*, 29–49. [**Google Scholar**]