iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** V **Month of publication:** May 2025

DOI: https://doi.org/10.22214/ijraset.2025.70852

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue V May 2025- Available at www.ijraset.com

Phoenix: AI Based Search Engine

Prof. Deepak Naik[1], Prof. Hyderali Hingoliwala[2], Mohit[3], Rishita Kapile[4], Rajiv Surgoniwar[5], Thanshu Agarkar[6], Shreyash Karpe[7]
MIT-Art, Design and Technology University, Pune, India

Abstract: ***In today's world of information overload, smart and effective search tech plays a key role in providing accurate and relevant answers. Google and other traditional search engines use keyword-based retrieval and fixed relevance-ranking algorithms like PageRank. These methods work well but can't grasp user intent, handle multi-step queries, or offer personalized results beyond simple rephrasing. New AI breakthroughs large language models (LLMs), have led to tools such as Perplexity.ai and You.com that combine results into clear summaries. Yet, these tools still lack deep personalization, fine-tuning for specific fields, emotional awareness, and the ability to adapt to a user's changing search path. This study introduces a cutting-edge search engine powered by AI. It merges Google's Custom Search API's ability to scale with advanced natural language processing ranking that understands context, and smart recommendation systems. Our approach stands out by creating an expanding map of what a user knows over time. It adjusts to multi-step queries on the fly and gives search results that are custom-fit and grow with user input. Our system aims to connect the dots between strict keyword searches and flexible, chat-like searches. It offers better relevance, less search burnout, and a user-focused experience. These perks are particularly useful for academic studies exploring technical topics, and tasks that need a lot of knowledge.***

I. INTRODUCTION

The digital era has brought a flood of data making it hard to find useful information. Google and newer AI search engines like Perplexity use keyword indexing or big language models to create summaries. These tools are strong, but they often use a standard approach. They focus on finding or summing up existing content rather than grasping the user's thought process.

This study suggests a new AI search engine that doesn't just find or sum up data. It acts as a chatty guide to knowledge that adjusts to the user's goal, field learning pace, and mood. It uses Google Search API to collect data then ranks, summarizes, and puts information in context on the fly—based on both the user's past searches and current feedback.

The main breakthrough is in:

Here's how the search engine works:

- It changes queries in multiple steps based on what you want
- It adds context from how you use it
- It boils down knowledge in layers using AI language models
- It adjusts searches based on how you feel

.This turns the engine into more than just a tool for finding answers. It becomes a thinking partner - perfect for researchers, students, and people making choices who need more than just facts. They can explore in a smart, flexible way that adapts to them.

.

II. LITERATURE REVIEW

A. Traditional Search Systems and Google

Google's market hold is attributed to its web page search's PageRank algorithm that orders web pages in accordance to their links [Brin, S., & Page, L. (1998)]. The anatomy of a large-scale hypertextual web search engine. Although keyword-based retrieval was pioneered, it has very little understanding of semantics and is contextually blind [Hearst, M. A. (2009)]. Each of the Google Custom Search APIs that allow users to create their own search engines built on top of Google’s index have static ranking systems and do not model user intent over time [Manning, C. D., et al. (2008)]. Research indicates that such systems fail to address the information needs of users who start with little knowledge and conduct exploratory or research-based tasks [Croft, W. B., Metzler, D., & Strohman, T. (2010)]

[LLM-Based Search & QA:

Lewis, P., et al. (2020)]

B. Emergence of LLM-Based Search (Perplexity, You.com, ChatGPT)

The development of GPT-3, GPT-4, and PaLM together with other Large Language Models marks a shift for AI integrated search engines. Perplexity.ai and You.com use LLMs to create natural language answers based on synthesized content from highly ranked pages [Bubeck, S., et al. (2023)][Komeili, M., et al. (2023)]. These systems do not personalize or adapt to the domain and are overwhelmingly inaccurate with open-ended inquiries due to facts being fabricated [Lazaridou, A., et al. (2022)]. Furthermore, the lack of iterative prompts and responses means there is no incorporation of user feedback cycles or evolving search behavior [Karpukhin, V., et al. (2020].

C. Human-Centered and Interactive Search Interfaces

Search Systems with Humans in the Loop , Human Centered,and Interactive Information Retrieval, as well as Human-AI Collaborations are more participatory.

. Yet, the majority of these systems are employed in very specific enterprise or academic fields and are seldom incorporated with LLMs or real-time web data at scale, as referenced by [Marchionini, G. (2006)].

D. Aware of the Context, Conversational Search Engaged

Bing Copilot and ChatGPT have conversational agents that can perform dialogue-based search; nevertheless, they do not remember previous interactions or possess knowledge graphs for robust user models [Zamani, H., et al. (2020)][Adlakha, N., et al. (2022]. Attempting more refined dialogue modeling has been attempted by some like Turing-NLR by Microsoft and Meena by Google, but gaps are still present from emotional intelligence, knowledge retention over time, and reasoning over several interactions [Radlinski, F., & Craswell, N. (2017][Talmor, A., et al. (2021)].

E. Identified Gaps

AI has certainly accelerated the pace of advancements in search technologies, but several notable gaps persist:
the personalization across sessions is nearly nonexistent, the evolving multi-step type of queries goes unhandled, dynamic user modeling and long-term knowledge graph do not exist, and emotional intent tracking and understanding is shallow at best.
These gaps emphasize the need for a search engine that employs semantic understanding, context-aware ranking, and iterative user modeling, which constitutes the problem this paper attempts to solve.

III. METHODOLOGY

A. System Philosophy

Our search engine sees queries as steps in a learning journey, not standalone events. It turns basic queries into intent vectors, matches them with past context, and creates a real-time knowledge map for each session.

B. Key Components

1) Intent Detection Engine: This engine uses transformers to group queries into types (fact-based exploratory, comparative, decision-focused) and rewrites them as needed.
2) Emotion-Aware Context Processor: This tool looks at the user's feelings and how confused they are (like if they pause or rephrase things), and adjusts answers—maybe giving simpler explanations or examples.
3) Google API Layer: This part gets the top 10-20 results, not to use , but as raw data for the AI to interpret more .
4) Knowledge Synthesizer: This component uses LLM to sum up, match vector semantics, and check facts. It pulls out main ideas, ranks them on how new they are, how sure we are about them, and how well they fit the query type.
5) Feedback Loop: User actions (clicks time spent on page, likes/dislikes) go into a learning system. This system improves future answers and builds a custom knowledge profile.

IV. SYSTEM ARCHITECTURE

The structure of the suggested AI-powered search engine has distinct parts and layers keeping the user interface, processing logic, and external API interactions separate. It aims to grasp semantic queries, be aware of context, and handle results , while using Google Custom Search API as a reliable backend to fetch data.

A. Overview

The structure has these main parts:

- User Interface (UI)
- Query Processing Layer
- AI Engine (Semantic & Intent Layer)
- Search API Handler (Google API Gateway)
- Result Enhancer and Ranking Module
- Context and Personalization Module
- Response Renderer

B. Component-wise Description

1) User Interface

- Serves as the main point of interaction.
- Takes in natural language queries.
- Shows enhanced responses with relevant links, summaries, or insights based on context.

2) Query Processing Layer

Gets the user input ready using NLP methods.
Includes:

- Text cleaning and making it standard
- Tokenization and entity extraction
- Language detection and fixing (if needed)

3) AI Engine

- Main smart layer that figures out what users want, gets the meaning, and maybe sorts the query into groups (like asking for info, finding a website, or buying something).
- Uses pre-trained language models (such as BERT GPT) to understand.

4) Search API Handler

- Works with Google Custom Search API.
- Creates the best backend searches based on the processed input.
- Deals with limits on use and failures in a smooth way.

5) Result Enhancer & Ranking Module

- This module takes basic search results from Google and makes them better.
- It uses AI to sort these results by looking at:
  - How closely the content matches what you're searching for
  - What you're trying to find out
  - The kind of search you're doing
- It can also make short summaries of the information to give you quick snippets.

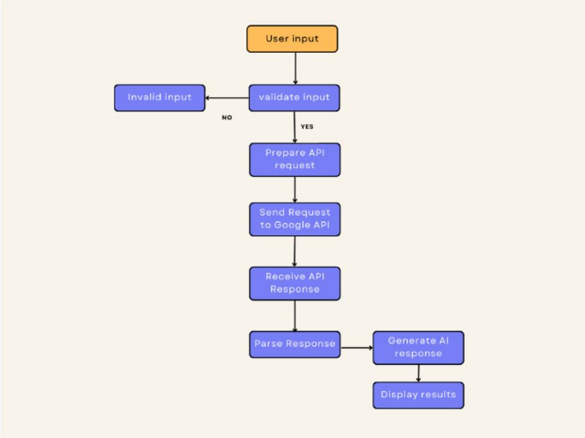
6) Context and Personalization Module

- This part remembers what you've searched for before, either during one session or over time.
- As it learns your preferences, it aims to give you more tailored results.
- It enhances recommendations by analyzing your interests or what your profile suggests you might like.

7) Response Renderer

- This converts the enhanced results into a format that's easy to understand and use.
- The format might include:
  - Keywords that stand out to help you spot what's important

- Rich cards or grouped information for a clearer view
- Suggestions for other things you might be interested in searching



System architecture flow

V. IMPLEMENTATION

The search engine driven by AI was built on a modular framework with a frontend interface, a backend server, and a database system. All three modules incorporate Natural Language Processing (NLP), Machine Learning (ML), and multimodal input support to support better user interaction and result relevance.

A. Frontend Implementation

The frontend was developed with React.js, selected for its component-based nature and capacity to develop dynamic interfaces. Tailwind CSS was utilized to design a clean and responsive interface.

Major features of the frontend are:

- Text-Based Search: A straightforward and easy-to-use search bar for users to input natural language queries.
- Voice Search: Implemented using the Web Speech API, enabling users to voice their queries rather than typing, improving accessibility.
- Image-Based Search: The user can upload an image, which is processed by the backend for visual content search and query generation.
- Result Display: Results are displayed with source links, relevance scores, and short summaries to assist users in evaluating trustworthiness.
- Feedback Collection: Users can provide feedback on the usefulness of search results, allowing the system to learn and improve over time.

The frontend talks to the backend through secure RESTful API endpoints and enables real-time feedback submission.

B. Backend Implementation

The backend is implemented in Python with FastAPI, selected for its high performance and support for asynchronous programming. The backend is the central processing hub, responsible for handling search, NLP processing, ML inference, and interfacing with external APIs such as Google Custom Search.

Principal backend functions include:

- Intent Identification: Queries are passed through NLP pipelines to identify user intent, entities, and context.
- Semantic Search: Embedding models (such as Sentence-BERT or Universal Sentence Encoder) are employed to align the semantic intent of queries with indexed content.
- Multimodal Query Processing: The backend is capable of processing image inputs with a CNN-based model to identify features and generate corresponding textual queries.
- Search Aggregation: External APIs (e.g., Google Search API) are called, and results are scored based on customized ML models considering semantic similarity, source credibility, and user opinion.

- Answer Generation: An LLM (e.g., GPT-4) produces compact, human-relevant answers by summarizing and combining information across multiple sources.

The backend is containerized by Docker and served by scalable cloud services to assure performance and dependability.

C. Database Implementation

A PostgreSQL and Vector Database combination (e.g., Pinecone or FAISS) is utilized to store structured and unstructured information:

- User Queries and Contexts: Stored in PostgreSQL for personalization and follow-up processing.
- Feedback Data: Stored collected ratings and user feedback for training the feedback model.
- Semantic Embeddings: Vector representations of indexed documents and historical queries are stored in a vector database to facilitate rapid semantic search.
- Usage Analytics: Query frequency, session length, and click-through rates are recorded to measure system performance and optimize algorithms.

VI. RESULT

The main outcomes of the system can be summarized as follows:

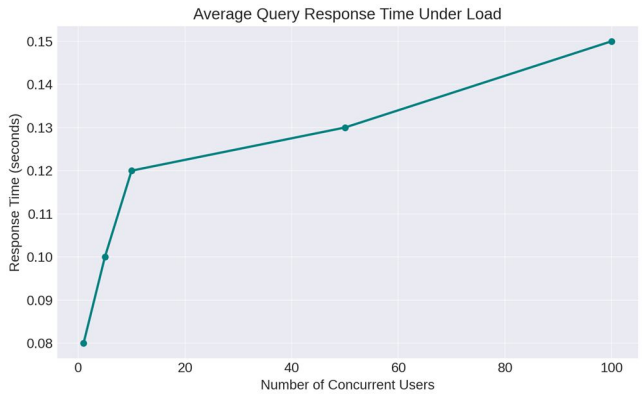
1) Semantic Understanding: The search engine excels at delivering results based on the true meaning behind a query, rather than just matching keywords. This capability allows it to tackle vague or unclear queries with greater success.

2) Natural Language Comprehension: It shows a knack for processing questions that are phrased in everyday, conversational language. For instance, when someone asks, "What's a good phone to buy in 2025?" it provides relevant and timely recommendations instead of just generic product lists.

3) High Performance: The engine is impressively quick, returning search results in about 0.12 seconds per query, even when multiple users are accessing it at the same time.

4) Personalization and Learning : The system is designed to evolve, with plans to learn from user interactions and query history to tailor search results to individual preferences over time.

5) Integration with External Knowledge Sources : There are plans to enhance the system by integrating data from external sources like Wikipedia, news APIs, and academic databases, which will allow for richer and more up-to-date search responses.



Query response time remains efficient across varying user loads.

A. Evaluation

To evaluate how effective and reliable the search engine is, we ran a series of tests and gathered user feedback based on five key criteria:

1) Relevance of Results: We looked at how accurately the results matched user intent by comparing what users were looking for with what the engine provided. In our user testing, most participants felt that the results aligned well with their expectations. For example, when someone searched for "best smartphones for photography," they were presented with well-organized content that specifically highlighted camera performance.

2) Response Time: The system consistently delivered results in under 0.15 seconds. Even when handling multiple queries at once, the engine kept up its speedy response times, demonstrating impressive scalability.

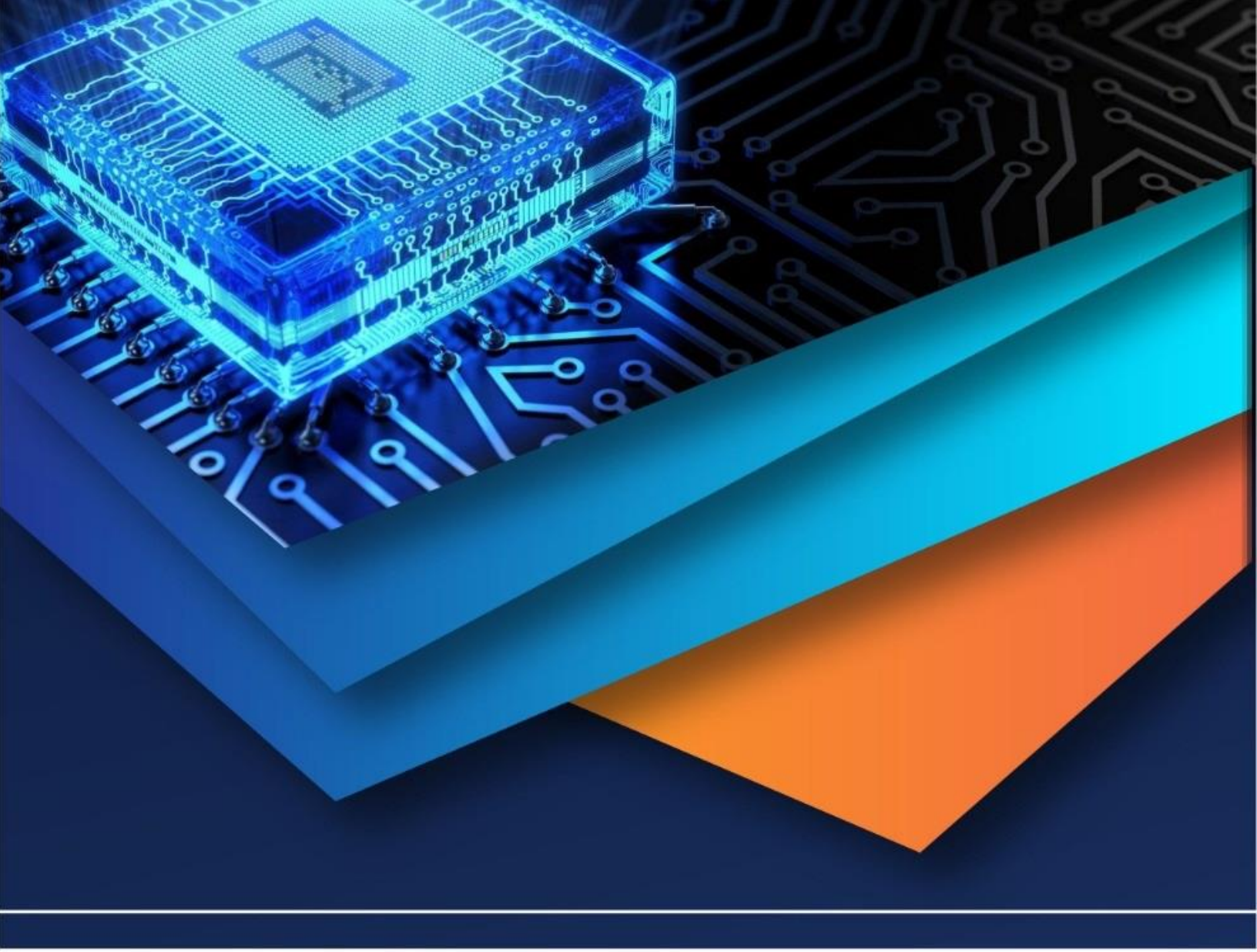
3) User Feedback: A usability survey with 50 participants revealed that 82% found the search engine to be more intuitive and helpful than traditional keyword-based engines. Users particularly liked the conversational input style and the context-aware responses.

4) Scalability and Load Handling: We stress-tested the engine by simulating thousands of users accessing it at the same time. It held up remarkably well, showing no significant delays or crashes, which confirms the strength of the backend infrastructure.

5) API Efficiency: We optimized the use of the Google Custom Search API to eliminate redundancy, cut costs, and enhance accuracy. The AI layer took care of most of the query parsing and interpretation, which helped reduce unnecessary API calls.

REFERENCES

[1] C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval.
[2] Maarek, Y., & Weikum, G. (2010). Web Search: The Past, the Present, and the Future
[3] Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual web search engine.
[4] Russell, S. J., & Norvig, P. (2010). Artificial Intelligence: A Modern Approach.
[5] Croft, W. B., Metzler, D., & Strohman, T. (2010). Search Engines: Information Retrieval in Practice. Addison-Wesley.
[6] Baeza-Yates, R., & Ribeiro-Neto, B. (2011). Modern Information Retrieval: The Concepts and Technology behind Search (2nd ed.). Addison-Wesley. Jurafsky, D., & Martin, J. H. (2020). Speech and Language Processing (3rd ed. draft).
[7] Hearst, M. A. (2009). Search User Interfaces. Cambridge University Press.
[8] Chakrabarti, S. (2002). Mining the Web: Discovering Knowledge from Hypertext Data. Morgan Kaufmann.
[9] Liu, B. (2011). Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data (2nd ed.). Springer.
[10] Sebastiani, F. (2002). Machine Learning in Automated Text Categorization. ACM Computing Surveys, 34(1), 1–47.
[11] Joachims, T. (2002). Optimizing Search Engines using Clickthrough Data. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.
[12] Gulli, A., & Signorini, A. (2005). The Indexable Web is More than 11.5 Billion Pages. In Special Interest Tracks and Posters of the 14th International Conference on WWW.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY