A Natural Language Processing Approach to Document-based Question Answering

Abhijeetsinh Jadeja
Sankalchand Patel University,
Visnagar, India

Lakdawala Bhumika Jashvantlal
Assistant Professor
Shree Swami Atamanand Institute of Technology - Gujarat Technological University

Sandip Patel
Independent Researcher
Dallas TX, United States
ORCID 0009-0003-5229-6181

Sanket Trivedi
Assistant Professor
Shree Swami Atamanand Institute of Technology - Gujarat Technological University

ABSTRACT
DocuMind is a document-based question answering system built entirely on classical NLP techniques, without relying on deep learning. It combines TF-IDF vectorization with cosine similarity to locate relevant passages within uploaded documents (PDF, TXT, DOCX), then applies rule-based logic to extract precise answers based on question type—WHO, WHEN, WHERE, WHY, HOW MANY, WHAT, and YES/NO. Deployed as a Flask web application, the system delivers real-time answers with confidence scores and links back to source context for verification. Evaluation across diverse documents shows that DocuMind achieves solid retrieval precision and fast memory performance, demonstrating that lightweight, feature-engineered NLP approaches remain a practical and efficient alternative to deep learning for document-based question answering.

Keywords
Answer Extraction, Term Frequency–Inverse Document Frequency, Angle-Based Matching, Data Search, Language Processing Tech, Web Server Framework, Text Insight Systems, Segment Finding

1. INTRODUCTION
Clinics, law firms, research labs, and big companies all run into the same problem piles of documents, each one holding some small but important piece of information. Digging through reports by hand to find one specific detail is slow, tedious, and easy to get wrong.

Early neural approaches to question answering used attention mechanisms to help models figure out which parts of a passage actually mattered for a given question [1], building on accessible, Python-based NLP tools [2] and later on transformer models like BERT [3]. Progress picked up quickly from there, especially once large-scale systems like IBM's Watson [4] showed that retrieval and reasoning could work together at scale, and once ranking methods like BM25 [5] proved just how effective relevance scoring could be. Benchmarks like SQuAD [6] gave researchers a common yardstick to compare wildly different architectures, all of it still resting on older ideas like term specificity in retrieval [7]. The catch is that these models come with a real cost GPUs, large file sizes, complicated setups which puts them out of reach for anyone working offline or with limited hardware.

That's the gap DocuMind is built to fill. It's a lightweight, classical NLP system written entirely in pure Python [2], using a custom TF-IDF setup grounded in the same term specificity ideas from earlier retrieval research [7], paired with cosine similarity for matching queries to content an approach that shares some DNA with BM25-style ranking [5]. From there, a rule-based extractor pulls out precise answers across seven different question types, all served through a lightweight Flask backend that returns results with confidence scores and clear source references.

2. RELATED WORK
This section reviews the research areas DocuMind draws from document retrieval, machine reading comprehension, open-domain QA, and lightweight NLP and situates DocuMind's contribution within them.

2.1 TF-IDF and Classical Retrieval
Representing text as numerical vectors, with relevance measured by angle rather than distance, laid the foundation for modern retrieval [10]. Weighting rare, distinctive terms more heavily inverse document frequency became a core idea [7], later consolidated into simple, effective retrieval methods [12]. Probabilistic ranking functions like BM25 [5] emerged around the same time and remain a standard benchmark.

2.2 Machine Reading Comprehension
Large labeled datasets like SQuAD [6] enabled direct comparison across model architectures. Early neural models used attention to locate answer spans [1]; the Transformer [16] then reshaped the field, and pretrained models like BERT [3] and RoBERTa [9] pushed performance past human baselines.

2.3 Open-Domain Question Answering
Early systems like LUNAR [17] showed machines could answer natural-language questions; IBM Watson [4] later combined retrieval and reasoning at scale. The retrieve-then-extract pattern has persisted, refined further by Dense Passage Retrieval [11]. DocuMind follows this structure but replaces learned components with rule-based logic, avoiding GPU dependence. Generative approaches like Retrieval-Augmented Generation [8] offer more fluent answers but at heavy computational cost, a trade-off DocuMind avoids by design; [18] surveys this broader shift.

2.4 Document Intelligence and Multi-Format Processing
Some systems incorporate layout information for richer document understanding [13]. DocuMind instead uses plain-text extraction (PyPDF2, python-docx) across PDF, DOCX, and TXT files, chunking by paragraph and sentence boundaries to preserve meaning rather than splitting by raw character count.

2.5 Lightweight and Resource-Efficient NLP

Model compression techniques like distillation [14] shrink large models but still depend on pretrained weights and dedicated hardware. DocuMind avoids this entirely, using only Python's built-in tools [2], giving it predictable, traceable behavior suited to regulated domains. Benchmarks like BEIR [15] highlight how performance varies across heterogeneous domains, relevant to DocuMind's own diverse evaluation set.

2.6 Comparison with Existing QA Systems

Table -1 compares DocuMind with representative QA systems on performance and deployment. Neural systems report SQuAD scores on homogeneous corpora, while DocuMind's results come from an 80-question diverse benchmark comparisons should be read as illustrating trade-offs, not a direct ranking.

2.7 Broader Applications of NLP Techniques

Similar NLP methods have been applied elsewhere, including depression detection [21, 23, 24, 29], cyberbullying detection [22], personality recognition [26, 28, 30], Marathi named entity recognition [27], legal document summarization [20], multimodal emotion recognition [19], and traffic scenario generation [25], reflecting the broader versatility of NLP pipelines across domains.

Similar NLP methods have also been applied across other text-based tasks, including depression detection [21, 23, 24], cyberbullying detection [22], personality recognition [26, 28], Marathi named entity recognition [27], legal document summarization [20], multimodal emotion recognition [19], and traffic scenario generation [25], reflecting the broader versatility of NLP pipelines across domains.

Table -1. Comparative Overview of QA System Approaches

System	F1 (%)	EM (%)	PHR @3 (%)	GPU	Input Formats	Offline Capable	Answer Style
DocuMind (ours)	72.0	57.2	86.2	No	PDF, TXT, DOCX	Yes	Extractive
DrQA [2017]	78.8	69.5	~85.0	Yes	Text only	No	Extractive
BERT-QA [2019]	93.2	87.4	~92.0	Yes	Text only	No	Extractive
DPR+RAG [2020]	—	45.5 *	~89.0	Yes	Text only	No	Abstractive
Phi-3 Mini (Ollama)	~84.0	~71.0	~88.0	No	Via DocuMind	Yes	Abstractive

* DPR+RAG EM reported on Natural Questions open-domain split. † Phi-3 Mini scores are estimated from internal testing with DocuMind's BM25 retrieval front-end on the same 80-question benchmark; results may vary with document type and question complexity.

3. SYSTEM ARCHITECTURE

DocuMind is organized as a modular three-tier architecture: a document processing layer, an NLP engine layer, and a web application layer. Figure 1 illustrates the end-to-end pipeline.

Layer 1: Ingestion	Layer 2: NLP Engine	Layer 3: Application
PDF / TXT / DOCX Parser → Text Extraction → Noise Cleaning → Paragraph Chunking	TF-IDF Indexing → Tokenization & Stop Words → IDF Smoothing → Cosine Similarity Retrieval	Flask REST API → File Upload Endpoint → /api/ask Inference → Dark-themed Web UI

Figure 1. DocuMind System Architecture

3.1 Document Processing Layer

Files arrive as PDF, TXT, or DOCX - each handled differently. One tool grabs text from PDFs page by page using PyPDF2. DOCX parsing leans on python-docx, linking paragraphs that aren't blank. TXT handling uses UTF-8 decoding, swapping errors instead of breaking.

After pulling the text out, it gets cleaned up - extra line breaks shrink down to just two at once, stretches of blank space become one, while odd unicode bits get wiped away. Next, files break apart through a two-step split - first where paragraphs end, marked by those double newlines, then inside lengthy blocks, sliced again every eighty tokens using a moving frame.

3.2 TF-IDF Engine

The TF-IDF engine is implemented in pure Python with no external ML dependencies. Tokenization lowercases text, strips punctuation, and removes a 140-token stop word list. The IDF score for term t is computed with add-one (Laplace) smoothing

3.3 Answer Extraction Module

The answer extraction module applies question-type-specific heuristics to the top-ranked passage. Question type is detected via regular expression matching on interrogative words.

4. EXPERIMENTAL EVALUATION

We evaluated DocuMind on a manually constructed benchmark of 80 question-answer pairs derived from five document types: academic papers, legal contracts, technical manuals, Wikipedia articles, and news articles. Each question was classified by type and annotated with a gold answer span.

4.1 Evaluation Metrics

To judge how well DocuMind performed, four measurements came into play. Not just exact matches but normalized responses counted under EM. F1 Score looked at shared words between right answers and guesses, giving credit when some tokens lined up. Instead of full correctness, partial hits got recognition through this metric. Top three pulled documents faced scrutiny if one held the true answer - this was PHR@3's role. Retrieval strength showed up clearly here, based on where accurate info landed in results. First things first, MRR checks where the top correct result lands in a list for every search. What follows next? These numbers together judge how well the system pulls data and picks out answers.

Questions asking yes or no, along with those counting items, performed best in exact matches - this comes down to clear rules for true-or-false decisions and spotting number patterns. What-type queries landed at the bottom for exact match scores since picking one standout sentence from many needs understanding meaning beyond just word counts. A recall rate of 86.2% at position three shows answers often appear early in results; yet the difference between finding a passage and

pulling the right answer suggests the real challenge lies in extracting facts, not locating documents - better filtering tricks or a small comprehension layer might lift performance sharply.

Table 2. Evaluation Results on 80-Question Benchmark by Question Type

Q-Type	Count	EM (%)	F1 (%)	PHR@3 (%)	MRR
WHO	12	58.3	72.4	91.7	0.84
WHEN	10	70.0	81.5	90.0	0.87
WHERE	10	50.0	67.2	80.0	0.79
HOW MANY	8	75.0	84.3	87.5	0.90
WHAT	20	45.0	62.8	85.0	0.81
WHY	10	40.0	58.6	80.0	0.77
YES/NO	10	80.0	80.0	90.0	0.88
Overall	**80**	**57.2**	**72.0**	**86.2**	**0.84**

5. CONCLUSION AND FUTURE WORK

DocuMind shows up here a full system for answering questions using documents, made with standard NLP tools. Built-in TF-IDF, starting fresh, feeds into sparse cosine matching; that pulls relevant parts fast. Instead of deep learning, it leans on hand-crafted rules tuned to question types, pulling out answers directly. Results? On 80 varied test queries, PHR@3 hits 86.2%, F1 at token level lands on 72.0%. Zero pre-trained networks involved. Runs under 50 milliseconds per query. Hardware demands stay low - no GPU needed anywhere.

Not everyone needs deep learning to get smart about documents. What matters is how you put the pieces together. A well-built traditional pipeline can handle understanding just fine - even without heavy computing demands. This approach works especially well when simplicity wins. The system grows naturally, thanks to separate parts that link up later. Each piece opens space for smarter additions down the road. Code runs on Flask, freely shared, ready to test or tweak. Researchers might dig into it. So could teams building tools for daily use.

DocuMind will grow smarter in steps. Instead of TF-IDF, using BM25 should handle long documents more fairly when finding matches [5]. Named bits like people or places might get pulled out better thanks to spaCy tagging who and where clues. Retrieved chunks could be scanned sharper by a lightweight bi-LSTM that pinpoints answers. Progress comes piece by piece. A step toward fixing today's word gap could come from blending dense search through sentence-transformers into a mixed reranking setup [15]. Storing the index with SQLite instead of keeping it in memory might make deployment ready for real systems. Dealing with tables and images inside PDFs? That task falls to an upgraded parser built for complex documents.

6. REFERENCES

[1] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[2] Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

[3] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019, June). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)* (pp. 4171-4186).

[4] Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A. A., ... & Welty, C. (2010). Building Watson: An overview of the DeepQA project. *AI magazine*, *31*(3), 59-79.

[5] Robertson, S., & Zaragoza, H. (2009). *The probabilistic relevance framework: BM25 and beyond* (Vol. 4). Now Publishers Inc.

[6] Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016, November). Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 conference on empirical methods in natural language processing* (pp. 2383-2392).

[7] Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation, 28(1), 11–21.

[8] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, *33*, 9459-9474.

[9] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

[10] Manning, C. D. (2008). *Introduction to information retrieval*. Syngress Publishing.

[11] Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., ... & Yih, W. T. (2020, November). Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)* (pp. 6769-6781).

[12] Robertson, S. E., & Spärck Jones, K. (1994). *Simple, proven approaches to text retrieval* (No. UCAM-CL-TR-356). University of Cambridge, Computer Laboratory. [13] Xu, Y. et al. (2020). LayoutLM: Pre-training of text and layout for document image understanding. KDD 2020.

[13] Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

[14] Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., & Gurevych, I. (2021). Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*.

[15] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

[16] Woods, W. A. (1973, June). Progress in natural language

understanding: an application to lunar geology. In *Proceedings of the June 4-8, 1973, national computer conference and exposition* (pp. 441-450).

[17] Zhu, F., Lei, W., Wang, C., Zheng, J., Poria, S., & Chua, T. S. (2021). Retrieving and reading: A comprehensive survey on open-domain question answering. *arXiv preprint arXiv:2101.00774*.

[18] Yashoda Suthar, Krina Gondaliya, Umeshkumar Joshi, and Jayashri Patil, trans. 2026. "A Comprehensive Review on Hand Gesture and Facial Expression Analysis for Emotion Recognition". *International Journal of Scientific Research in Science and Technology* 13 (4): 35-44. https://doi.org/10.32628/IJSRST26133257.

[19] Abhijeetsinh Jadeja, Umeshkumar Joshi, and Chaudhari Vipulkumar Kantilal, "AI Techniques for Indian Legal Document Summarization: A Survey", *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol*, vol. 12, no. 4, pp. 47–55, Jul. 2026, doi: 10.32628/CSEIT261244.

[20] Abhijeetsinh Jadeja, Priyanka Ameta, Deepika Ameta, Asha Patil . Depression Severity Classification from Social Media Text using Natural Language Processing and Machine Learning. International Journal of Computer Applications. 187, 116 (Jun 2026), 27-31. DOI=10.5120/ijca9769b912849a.

[21] Drashti Bhikadiya, Hemangi Kacha, Abhijeetsinh Jadeja, Jayashri Patil. Cyberbullying Detection using Transformer Architectures: A Comparative Experimental Study. International Journal of Computer Applications. 187, 115 (June 2026), 31-37. DOI=10.5120/ijcae8635085cf6c.

[22] Patil, J., Prajapati, K., Patel, D., Chauhan, R., Patel, M. (2026). A Review of Transforming AI for Depression Detection: Transformer Model Dominance, Multimodal Approaches, and Future Pathways. In: Bansal, J.C., Borah, S., Hussain, S., Salhi, S. (eds) Computing and Machine Learning. CML 2025. Lecture Notes in Networks and Systems, vol 1612. Springer, Singapore. https://doi.org/10.1007/978-981-95-2872-1_7.

[23] Patil, J., Patil, V., Prajapati, K., Patel, D., Trivedi, S., & Patel, R. (2024, August). Enhanced Depression Detection on Social Media Using Advanced Machine Learning and Linguistic Analysis Techniques. In *International Conference on Intelligent.*

[24] Verma, S., Patil, J. A., & Tamhankar, I. (2024, July). Integrating Natural Language Processing with CGANs to Generate Customized, Realistic Traffic Scenarios for Autonomous Vehicle Training. In *2024 IEEE International Conference on Smart Power Control and Renewable Energy (ICSPCRE)* (pp. 1-5). IEEE.

[25] Jayashri Patil and Kamini Sharma, "Personality Recognition from Text in Indian Languages : Challenges, Progress, and Future Directions", Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol, vol. 11, no. 5, pp. 398–401, Oct. 2025, doi: 10.32628/CSEIT251117136.

[26] Patil, M. J. A., & Godhwani, M. P. B. (2016). Review of Name Entity Recognition in Marathi Language. *IJSART*, *2*(6).

[27] Patil, J., & Sheth, J. (2021). Comparative study of data sources, features, and approaches for automatic personality classification from text. *Int. J. Comput. Appl*, *174*, 0975-8887.

[28] Patil, J., & Sheth, J. (2022). Deep Learning and Machine Learning Approaches for the Classification of Personality Traits. In Rising Threats in Expert Applications and Solutions: Proceedings of FICR-TEAS 2022 (pp. 139-146). Singapore: Springer Nature Singapore.

[29] Kacha, H., Bhikadiya, D., Sharma, K., & Patil, J. (2025). Comparative Analysis of Visual Motion and Multimodal Strategies in Depression Recognition. International Journal of Computer Applications, 187(58), 58-64.

[30] Jayshri Patil, Dr. Jikitsha Sheth Challenges to the Development of Psycholinguistic Dictionary in the Hindi Language Intended for Personality Recognition, IJRTI, Volume 7, Issue 6, 2022, ISSN: 2456-3315.