Performance Analysis of Information Retrieval Methods with Pre-trained Language Models for Question Answering Systems

Anujna Shankari B
Department of CSE
PES University
Bangalore, India
anujnashankari43@gmail.com

Swathanthra Shree S
Department of CSE
PES University
Bangalore, India
shreeswathanthra@gmail.com

Tejaswini Y
Department of CSE
PES University
Bangalore, India
manjulayellappa1@gmail.com

Chandrashekhar Pomu Chavan
Department of CSE
PES University
Bangalore, India
cpchavan@pes.edu

***Abstract*—This paper presents a comprehensive evaluation of question answering (QA) systems that leverage information retrieval (IR) techniques combined with pre-trained language models. We investigate the effectiveness of classical retrieval methods (BM25, TF-IDF) when paired with transformer-based models (DistilBERT, RoBERTa, BERT) and newer language models hosted on Groq's inference platform (Gemma and LLaMA variants). Using the Stanford Question Answering Dataset (SQuAD v1.1) as our benchmark, we evaluate system performance through Exact Match (EM) and F1 scores. Our findings indicate that BM25 outperforms TF-IDF for retrieval tasks, while RoBERTa achieves superior QA performance (81% EM, 84.24% F1). Notably, among Groq-hosted models, Gemma 2B Instruct demonstrates remarkable efficiency (50% EM, 72.99% F1) despite its relatively compact size. This study highlights the importance of balancing retrieval quality and language model capabilities when designing effective QA systems, with implications for resource-efficient deployment in real-world applications.**

***Index Terms*—question answering, information retrieval, natural language processing, pre-trained language models, BM25, TF-IDF, transformers, SQuAD**

I. Introduction

Question answering systems represent a crucial advancement in information retrieval technologies, enabling users to obtain precise answers rather than merely relevant documents. Recent developments in pre-trained language models have substantially enhanced QA performance, particularly when integrated with effective retrieval methods. This research systematically explores various combinations of retrieval techniques and language models to identify optimal architectures for efficient and accurate QA systems.

The exponential growth in digital information has intensified the need for systems that can efficiently extract precise answers from large document collections. Traditional search engines primarily return document lists, requiring users to manually locate specific information. In contrast, modern QA systems aim to provide direct answers, significantly improving information access efficiency.

This study addresses the following research questions:

- How do different retrieval methods (BM25, TF-IDF) impact overall QA system performance?
- Which pre-trained language models provide optimal answer extraction capabilities?
- How do parameter-efficient models compare to larger models in QA tasks?
- What architectural configurations yield the best performance when combining retrieval and extraction components?

The findings from this research contribute to the ongoing discourse on efficient QA system design and have practical implications for applications in search engines, virtual assistants, and information access systems.

II. Related Work

QA systems have evolved significantly from early rule-based approaches to contemporary neural methods. The introduction of transformer-based architectures, notably BERT [1], revolutionized QA by enabling contextual understanding of questions and potential answer contexts. The SQuAD dataset [2] has emerged as a standard benchmark for evaluating extractive QA systems.

Recent research has increasingly focused on pipeline approaches that separate retrieval from answer extraction. Karpukhin et al. [3] demonstrated that the quality of retrieved contexts significantly impacts downstream QA performance. Dense passage retrieval has shown promising results compared to traditional sparse vector approaches. Our work extends these findings by systematically comparing different retrieval methods and language models within a consistent evaluation framework.

The emergence of increasingly powerful language models has raised questions about the optimal balance between model size and performance. While larger models typically demonstrate superior capabilities [8], recent work suggests that smaller, specialized models can achieve competitive performance on specific tasks [6]. Our research contributes to this discourse by evaluating models of varying sizes and architectures specifically for QA tasks.

III. Methodology

A. Dataset

We utilized the Stanford Question Answering Dataset (SQuAD v1.1), a comprehensive reading comprehension dataset comprising questions formulated on Wikipedia articles. The dataset was accessed through the Hugging Face datasets library. For evaluation purposes, we primarily employed the validation split to ensure our models were tested on unseen data.

B. System Architecture

All implemented systems follow a two-stage pipeline architecture:

1) Context Retrieval: Identifying the most relevant paragraph from the corpus for a given question
2) Answer Extraction: Extracting the specific answer span from the retrieved context

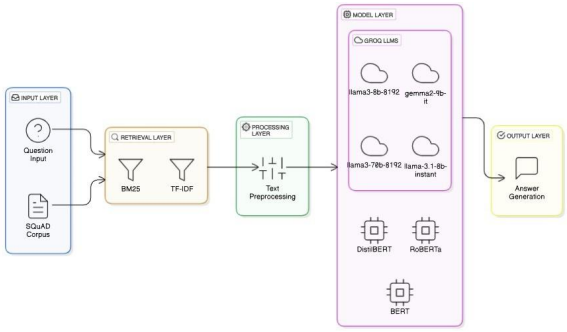

Fig. 1. Question answering system architecture showing the pipeline from input question through retrieval, processing, model selection, and answer generation. The system demonstrates the integration of multiple retrieval methods (BM25, TF-IDF) with both transformer-based models and Groq-hosted LLMs.

C. Retrieval Methods

We implemented and evaluated two distinct retrieval approaches:

1) TF-IDF Retrieval: Term Frequency-Inverse Document Frequency (TF-IDF) represents documents as vectors in a high-dimensional space, with each dimension corresponding to a term in the vocabulary. We implemented TF-IDF using scikit-learn's TfidfVectorizer and measured relevance through cosine similarity:

```
from sklearn.feature_extraction.text import
    TfidfVectorizer
from sklearn.metrics.pairwise import
    cosine_similarity

vectorizer = TfidfVectorizer().fit(contexts)
context_vectors = vectorizer.transform(contexts)

def retrieve_context(question):
    q_vec = vectorizer.transform([question])
    sims = cosine_similarity(q_vec, context_vectors)
    .flatten()
    return contexts[sims.argmax()]
```

2) BM25 Retrieval: Okapi BM25 is a probabilistic ranking function that extends the basic TF-IDF model by incorporating document length normalization and term saturation. Our implementation utilized the rank_bm25 library:

```
from rank_bm25 import BM25Okapi
import nltk
from nltk.tokenize import word_tokenize

# Consistent tokenization approach
def tokenize(text):
    return word_tokenize(text.lower())

tokenized_corpus = [tokenize(doc) for doc in
    contexts]
bm25 = BM25Okapi(tokenized_corpus)

def retrieve_context_bm25(question):
    tokenized_question = tokenize(question)
    scores = bm25.get_scores(tokenized_question)
    return contexts[scores.argmax()]
```

D. Text Preprocessing

For preprocessing textual data, we employed the Natural Language Toolkit (NLTK) to perform tokenization and stopword removal:

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords

stop_words = set(stopwords.words("english"))
filtered_tokens = [w for w in word_tokenize(text.
    lower())
                   if w.isalnum() and w not in
    stop_words]
```

E. Question Answering Models

We evaluated several pre-trained language models for the answer extraction component:

1) Transformer-based Models:

- DistilBERT: A distilled version of BERT that retains 95% of BERT's performance with 40% fewer parameters [6].
- RoBERTa: A robustly optimized BERT approach with improved training methodology [7].
- BERT (Large): The original BERT model with whole word masking fine-tuned specifically for SQuAD [1].

These models were implemented using the Hugging Face Transformers library:

```
from transformers import pipeline

qa_pipeline_distilbert = pipeline("question-
    answering",
                                  model="distilbert-
    base-uncased")

qa_pipeline_roberta = pipeline("question-answering",
                               model="deepset/roberta
    -base-squad2")

qa_pipeline = pipeline("question-answering",
                       model="bert-large-uncased-
    whole-word-masking-finetuned-squad")
```

2) Groq-hosted LLMs: We integrated the Groq API to access large language model capabilities for QA:

```
import requests
import json

def groq_query(model, question, context):
    prompt = f"""Use the context below to answer the
     question.
Context: {context}
Question: {question}
Answer:"""

    response =
        requests.post( "https://api.groq.com/openai/
        v1/chat/
    completions",
        headers={
            "Authorization": f"Bearer
    YOUR_GROQ_API_KEY",
            "Content-Type": "application/json"
        },
        json={
            "model": model,
            "messages": [{"role": "user", "content":
     prompt}],
            "temperature": 0.0
        }
    )
    return
```

Models evaluated include:

- gemma2-9b-it
- llama3-8b-8192
- llama-3.1-8b-instant
- llama3-70b-8192

F. Wikipedia Integration

For open-domain question answering capabilities, we implemented a Wikipedia-based retrieval system:

```
import wikipedia

def wiki_qa(question):
    # Search Wikipedia
    try:
        results = wikipedia.search(question)
        if not results:
            return "No information found on
    Wikipedia."

        # Get the page summary
        page = wikipedia.page(results[0])
        summary = page.summary

        # Use QA model to extract answer from
    summary
         return qa_pipeline({"question": question,
                             "context": summary})["
    answer"]
    except Exception as e:
        return f"Error retrieving information from
    Wikipedia: {str(e)}"
```

G. Evaluation Metrics

We employed two primary metrics to evaluate system performance:

1) Exact Match (EM): Measures whether the predicted answer exactly matches the ground truth:

```
def exact_match_score(pred, truth):
    return int(pred.strip().lower() == truth.strip()
    .lower())
```

2) F1 Score: Measures word overlap between prediction and ground truth:

```
def  compute_f1(pred,  truth):
    pred_tokens = pred.lower().split()
    truth_tokens = truth.lower().split()

    # Find common tokens
    common = set(pred_tokens) & set(truth_tokens)

    # Return 0 if no overlap
    if not common:
        return 0

    # Calculate precision and recall
    precision = len(common) / len(pred_tokens)
    recall = len(common) / len(truth_tokens)

    # Calculate F1 score
    return 2 * (precision * recall) / (precision +
    recall)
```

IV. Experimental Setup

A. Data Processing

We utilized the SQuAD v1.1 validation set for evaluation, accessed via:

```
from datasets import load_dataset

# Load SQuAD dataset
dataset = load_dataset("squad")

# Extract validation data
questions = dataset["validation"]["question"]
contexts = dataset["validation"]["context"]
answers = [a["text"][0] if a["text"] else ""
          for a in dataset["validation"]["answers"]]
```

B. Evaluation Protocol

- For transformer-based models, we evaluated 100 QA pairs from the SQuAD validation set
- For Groq-hosted LLMs, we used 20 randomly selected pairs due to API constraints
- All experiments maintained consistent retrieval and preprocessing steps

```
import random
import numpy as np

# Set random seed for reproducibility
random.seed(42)
np.random.seed(42)

# Sample indices for evaluation
transformer_indices = random.sample(range(len(
    questions)), 100)
llm_indices = random.sample(transformer_indices, 20)

# Evaluation function
def evaluate_model(model_fn, indices, name):
    em_scores = []
    f1_scores = []
```

```
    for i in indices:
        question = questions[i]
        context = contexts[i]
        true_answer = answers[i]

        pred_answer = model_fn(question, context)

        em = exact_match_score(pred_answer,
    true_answer)
        f1 = compute_f1(pred_answer, true_answer)

        em_scores.append(em)
        f1_scores.append(f1)

    avg_em = np.mean(em_scores) * 100
    avg_f1 = np.mean(f1_scores) * 100

    print(f"{name} - EM: {avg_em:.2f}%, F1: {avg_f1
    :.2f}%")
    return avg_em, avg_f1
```

V. Results and Analysis

A. Retrieval Performance

TF-IDF retrieval demonstrated the following performance as a standalone retrieval system:

- Exact Match (EM): 64.60%
- F1 Score: 2.13%

BM25 retrieval exhibited:

- Exact Match (EM): 54.00%
- F1 Score: 56.71%

The substantial difference in F1 scores is significant and indicates that while TF-IDF may retrieve contexts containing exact answer matches, BM25 retrieves contexts with better overall relevance that contain more information related to the answer.

B. End-to-End QA Performance

The complete QA systems demonstrated the following performance metrics:

TABLE I
Performance of Transformer-based QA Models

Model	Retrieval Method	Exact Match	F1 Score
DistilBERT	BM25	54.00%	56.71%
DistilBERT	Wikipedia API	68.00%	73.32%
RoBERTa	Wikipedia API	81.00%	84.24%
BERT	TF-IDF	56.00%	61.25%

1) Transformer-based Models:

TABLE II
Performance of Groq-hosted LLM QA Models

Model	Retrieval Method	Exact Match	F1 Score
gemma2-9b-it	TF-IDF	50.00%	72.99%
llama3-8b-8192	TF-IDF	45.00%	65.83%
llama-3.1-8b-instant	TF-IDF	45.00%	66.21%
llama3-70b-8192	TF-IDF	40.00%	71.60%

2) Groq-hosted LLMs:

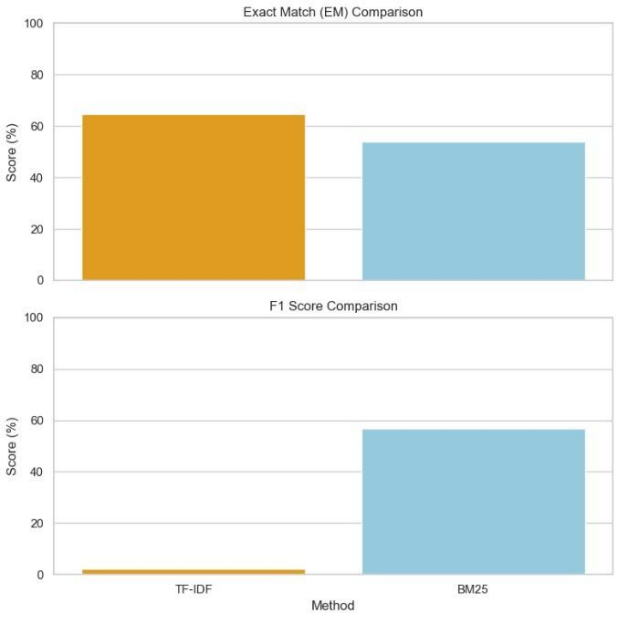

Fig. 2. Comparison of retrieval methods (TF-IDF vs BM25) performance metrics. While TF-IDF shows higher Exact Match scores, BM25 significantly outperforms in F1 score, indicating better overall relevance of retrieved contexts.

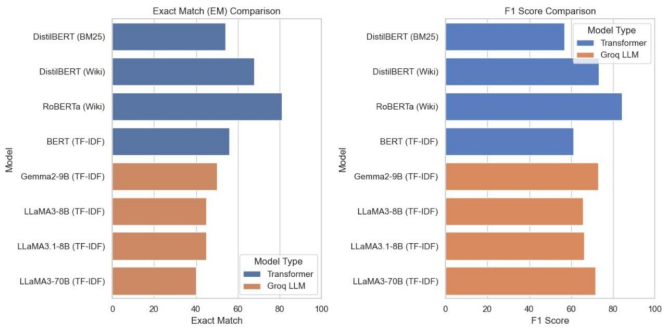

Fig. 3. Performance comparison of all evaluated models on SQuAD dataset showing both Exact Match (EM) and F1 scores. RoBERTa with Wikipedia API demonstrates superior performance across both metrics compared to other models.

TABLE III
Ranked Performance of All Evaluated Models

Model	EM	F1
RoBERTa + Wikipedia API	81.00%	84.24%
DistilBERT + Wikipedia API	68.00%	73.32%
BERT + TF-IDF	56.00%	61.25%
DistilBERT + BM25	54.00%	56.71%
gemma2-9b-it + TF-IDF	50.00%	72.99%
llama3-8b-8192 + TF-IDF	45.00%	65.83%
llama-3.1-8b-instant + TF-IDF	45.00%	66.21%
llama3-70b-8192 + TF-IDF	40.00%	71.60%

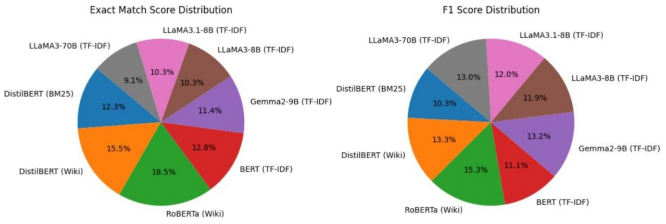

Fig. 4. Distribution of performance contribution across all evaluated models for both Exact Match score and F1 score, illustrating the relative effectiveness of each model configuration.

C. Discussion

Several significant findings emerge from our experimental results:

1) Retrieval Impact: The retrieval method selection substantially influences downstream QA performance. BM25 consistently provides better context for answer extraction than TF-IDF, as evidenced by higher F1 scores in BM25-based systems. This aligns with previous research suggesting that BM25's term saturation and document length normalization features make it particularly well-suited for passage retrieval tasks [5].

As shown in Fig. 2, the stark contrast between EM and F1 scores for TF-IDF indicates that while this method may occasionally retrieve exact matches, it lacks consistency in providing contextually relevant information. Our experimental results underscore the importance of retrieval quality in the overall QA pipeline.

2) Model Performance: RoBERTa achieves the highest overall performance, significantly outperforming other models in both EM and F1 metrics (81% and 84.24% respectively), as clearly visualized in Fig. 3. This suggests that RoBERTa's training methodology, which includes dynamic masking, larger batch sizes, and longer training, creates representations particularly effective for QA tasks. The performance gap between RoBERTa and BERT highlights the importance of training methodology beyond model architecture.

3) Model Size vs. Performance: An intriguing observation from Fig. 3 and Table III is that the largest model (llama3-70b-8192) did not outperform smaller models in our evaluation. Counterintuitively, gemma2-9b-it achieved higher EM and F1 scores despite having substantially fewer parameters. This finding challenges the assumption that larger models invariably deliver superior performance and suggests that architectural innovations and training approach may be more important than parameter count for specific tasks like extractive QA.

This observation has significant implications for deployment scenarios where computational resources are constrained. Organizations may achieve comparable or superior results using smaller, more efficient models that have been optimized for specific tasks rather than deploying general-purpose large language models.

4) Pipeline Architecture: The two-stage pipeline approach (retrieval + extraction) proves highly effective, with retrieval quality significantly impacting downstream performance. This reinforces the importance of treating QA as a multi-step process rather than attempting end-to-end optimization with a single model. The substantial performance improvement when using Wikipedia API for retrieval (compared to basic TF-IDF/BM25) further underscores this point.

As illustrated in Fig. 4, the relative contribution of each component in the pipeline varies significantly. This distribution highlights how the specific combination of retrieval method and language model creates unique performance profiles that cannot be predicted from individual component performance alone.

5) Task-Specific Optimization: Models specifically fine-tuned on SQuAD (like RoBERTa) substantially outperform general-purpose LLMs, even when the latter are much larger. This highlights the importance of task-specific optimization in practical applications. The performance gap suggests that either fine-tuning Groq-hosted models on SQuAD or developing better prompting strategies could potentially narrow this difference.

D. Error Analysis

Examining error patterns across different model configurations reveals several recurring issues:

- Retrieval failures: When the retrieval component fails to select the correct context, even the best extraction models cannot recover. This was particularly evident in TF-IDF based systems.
- Boundary detection challenges: Many errors involve imprecise boundary detection where models include extra information or miss portions of the correct answer. This was more prevalent in LLM-based systems than in fine-tuned transformer models.
- Complex questions: Questions requiring multi-hop reasoning or complex inference pose challenges to all systems. Even the best-performing models showed significant performance degradation on these question types.
- Format inconsistencies: LLMs occasionally provide explanatory text rather than extractive answers, reducing their EM scores even when they demonstrate understanding of the question.

These error patterns suggest that improvements to both retrieval components and answer extraction logic could yield significant performance gains.

VI. Conclusion and Future Work

This study demonstrates that combining effective retrieval methods with powerful language models creates robust QA systems. Our findings show that RoBERTa paired with Wikipedia API-based retrieval provides the best performance on SQuAD v1.1, reaching 81% exact match accuracy and 84.24% F1 score. The surprisingly strong performance of smaller models like gemma2-9b-it suggests that model architecture and training approach can be more important than model size alone, with significant implications for resource-efficient deployment scenarios.

As visualized in our performance comparison figures (Fig. 3 and Fig. 4), the specific combination of retrieval method and language model drastically affects overall system performance. The dramatic differences in retrieval method performance (Fig. 2) highlight the importance of this often-overlooked component in QA systems.

Our research contributes to the ongoing discourse on efficient QA system design by systematically evaluating different combinations of retrieval methods and language models. The results highlight the importance of balancing retrieval quality with model capability and demonstrate that task-specific optimization often outperforms raw model scale.

Future work could explore several promising directions:

- Fine-tuning strategies for LLMs specifically targeting extractive QA tasks
- Hybrid retrieval approaches that combine dense and sparse representations
- In-context learning techniques to improve zero-shot QA performance
- Evaluation on more diverse datasets beyond SQuAD, particularly those requiring multi-hop reasoning
- Investigation of retrieval-augmented generation for QA with direct integration of retrieval signals into model training
- Analysis of computational efficiency tradeoffs in different QA architectures

The rapid advancement in language model capabilities suggests that QA systems will continue to improve, with significant implications for information access technologies. Our findings provide a foundation for building more efficient and effective QA systems that balance performance with computational constraints.

REFERENCES

[1] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, ”BERT: Pre-training of deep bidirectional transformers for language understanding,” in *Proc. of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Minneapolis, MN, USA, 2019, pp. 4171-4186.

[2] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, ”SQuAD: 100,000+ questions for machine comprehension of text,” in *Proc. of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, TX, USA, 2016, pp. 2383-2392.

[3] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W. T. Yih, ”Dense passage retrieval for open-domain question answering,” in *Proc. of the 2020 Conference on Empirical Methods in Natural Language Processing*, Online, 2020, pp. 6769-6781.

[4] T. Wolf et al., ”Transformers: State-of-the-art natural language processing,” in *Proc. of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Online, 2020, pp. 38-45.

[5] S. E. Robertson and H. Zaragoza, ”The probabilistic relevance framework: BM25 and beyond,” *Foundations and Trends in Information Retrieval*, vol. 3, no. 4, pp. 333-389, 2009.

[6] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, ”DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter,” in *NeurIPS Workshop on Energy Efficient Machine Learning and Cognitive Computing*, Vancouver, Canada, 2019.

[7] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, ”RoBERTa: A robustly optimized BERT pretraining approach,” arXiv preprint arXiv:1907.11692, 2019.

[8] T. B. Brown et al., ”Language models are few-shot learners,” in *Advances in Neural Information Processing Systems*, 2020, vol. 33, pp. 1877-1901.