Efficient and verifiable responses using Retrieval Augmented Generation (RAG)

Henry Liang, Yu Zhou, and Vijay K. Gurbani
hliang@vailsys.com,yzhou@vailsys.com,vgurbani@vailsys.com

ABSTRACT

The rise of large language models (LLMs) like ChatGPT has greatly enhanced the efficiency of everyday tasks through automation. However, the deployment of LLMs for tasks such as responding to Request-for-Proposals (RFPs) is hindered by deficiencies like hallucinations and lack of response provenance. For such tasks, the aim of an automated response is to generate precise answers that can still be quickly reviewed and corrected by a human; therefore it is critical to optimize the system such that relevant source document sections are identified for as many questions as possible, and all relevant contexts are attributed correctly; this makes LLMs alone insufficient for this task. We present an improved Retrieval Augmented Generation (RAG) architecture for automated RFP completion that enhances relevant content generation and significantly reduces manual effort in drafting responses. The proposed improvements are two-fold: we present a novel text embedding scheme that combines a dense contextual embedding with a sparse statistical embedding for document retrieval, and we improve on the provenance of the generated response by presenting an algorithm that accurately provides the document page numbers as references when generating the answers. The practical deployment of this solution highlights its potential for automatic RFP completion, as well as its ability to act as an architecture for applications in various domains with differing complexity levels, especially when efficiency, accuracy, and verifiable responses are paramount.

KEYWORDS

LLM, RAG, RFP, hybrid retrieval, hybrid embedding

1 INTRODUCTION

Recent works have started to explore the augmentation of large-language models (LLMs) with localized data sources to generate relevant and provable answers to user queries. Such Retrieval Augmented Generation (RAG) systems [6] show superior performance compared to using only a pre-trained model [1]. While LLMs have demonstrated remarkable capabilities, their application in specialized enterprise tasks — such as automatically responding to Request-for-Proposals (RFPs) — still faces notable challenges. RFP is a critical application and the LLM response needs to be efficient, free of hallucinations, accurate, and verifiable as it establishes a contract between an enterprise and a customer. To address these challenges, this paper introduces an improved RAG architecture designed to automate the RFP completion process more effectively. The RFP application is a good use case for RAG because it is very important to ensure that LLMs do not generate unpredictable responses; not only are these responses customer-visible, but more importantly, these responses can be legally binding and may form the basis of a contract. Many enterprises have archival RFPs that can be mined for contextual knowledge on how to answer the target question. Our architecture does not eliminate human-in-the-loop, but it dramatically reduces the time required for a human to review – and correct – the answers, compared to answering each question manually[1].

Contributions: Concretely, this paper makes two novel contributions:

(1) We combine a dense contextual embedding (e.g., BAAI General Embedding (BGE embedding) [11]) with a sparse statistical embedding (e.g., Term Frequency-Inverse Document Frequency (TF-IDF) embedding [9]), to produce a hybrid embedding for efficient retrieval. It is demonstrated that this hybrid retrieval is superior compared to dense retrieval in isolation.
(2) The provenance of an answer generated by the LLM through the RAG process is established by providing the reference to a source attribute (certain page in a document) from which the answer was synthesized; the complexity here lies in reconstructing the document and identifying a page after the document has been pre-processed and split into chunks.

This system is *efficient* because the hybrid model produces enhanced embeddings that reduce no-context cases (i.e., cases where the information being sought is not found in the documents retrieved), and increases efficiency; it is *verifiable* because the source attribution to each generated response is annotated, allowing the human-in-the-loop to quickly ascertain the veracity of a generated answer. The techniques and architecture described in this paper can be used for any enterprise application that demands high levels of precision and accountability. The code developed in this work is publicly available at https://github.com/vail-systems/RAILS-AI-ML-2024.

The rest of this paper is organized as follows: Section 2 positions our contributions in the context of existing work on using RAG in LLM; Section 3 presents the components used to construct the enhanced RAG system, and Section 4 enumerates the datasets and the experimental setup. Section 5 presents a discussion on the results; we conclude in Section 6.

2 RELATED WORK

Various studies have explored enhancing retrieval mechanisms to improve performance on knowledge-intensive tasks. Guu et al.

[1]While the number of questions asked in an RFP vary, we have observed RFPs containing 400-800 questions.

AI-ML Systems Conference'24, October 2024, Baton Rouge, LA, USA

ACM ISBN 979-8-4007-1161-9…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

[2] introduced Retrieval-Augmented Language Model Pre-Training (REALM), which trains the retriever using a performance-based signal from unsupervised text. However, this method is computationally expensive. Adapting the model to a proprietary document corpus requires retraining to leverage the specific knowledge contained in those documents. Additionally, if the corpus is continually changing, the benefits of previous training efforts can be diminished, requiring frequent updates to the retriever. This can further increase complexity, whereas the hybrid retrieval method introduced in this paper does not require additional training.

Ramos [7] examined the application of TF-IDF to identify words in a corpus that are most relevant for use in queries. The study demonstrated that the TF-IDF algorithm excels in identifying and prioritizing words that are unique or relatively uncommon in the entire corpus yet frequent in specific documents. However, it exhibits limitations in addressing synonyms and word variations and capturing semantic relationships between words. These shortcomings could significantly diminish its effectiveness when applied to larger document collections. The hybrid retrieval method proposed in this paper builds upon the TF-IDF algorithm by integrating it with an embedding that captures the semantic relationships between words, circumventing some of the major limitations of TF-IDF.

Furthermore, Jalilifard et al. [3] proposed a method called Semantic Sensitive TF-IDF (STF-IDF), which enhances the traditional TF-IDF approach by incorporating semantic information from word embeddings. The STF-IDF method iteratively adjusts TF-IDF scores based on the cosine similarity between word2vec embeddings and the context of the most relevant terms. However, the word2vec model only provides static, context-independent word embeddings suitable for simpler tasks where individual word meanings are sufficient. The hybrid retrieval method builds on this approach by using a more sophisticated embedding model (BGE). Utilizing contrastive learning and transformer architecture, the BGE embeddings capture deeper semantic nuances, rendering it highly suitable for applications such as responding to RFPs, which demand a deeper understanding of textual context and semantics [11].

3 ARCHITECTURE

The architecture (Figure 1) of the RFP completion system improves upon the RAG architecture with the two main contributions of this paper: the hybrid retriever, and the Document Page Finder. This system is structured into four distinct yet interconnected components: a retriever, a vector database, a generator, and the Document Page Finder, each playing a critical role in the system.

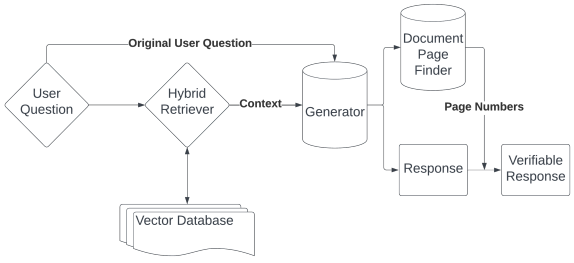

Figure 1: Architecture Diagram

3.1 Hybrid Retriever

The hybrid retriever component takes the user's question to query the vector database, where documents are stored as high-dimensional vectors representing semantic embeddings. It surfaces the most relevant documents by employing cosine similarity scoring, which measures the angle between the query vector and each document vector. This effectively gauges semantic similarity, ensuring that the retrieved documents are contextually aligned with the user's intent. After computing similarity scores, the retriever ranks all documents by relevance and selects the top five documents, which are returned as context for the final response generation.

The process of generating these embeddings for retrieval is a critical step that significantly influences the efficiency and accuracy of the retriever's operations. In this paper, two embedding schemes are compared. The first is a dense contextual embedding, represented by the output of the bge-large-en-v1.5 model (BGE [11]), which is specifically designed to capture deep semantic meanings from text data. This model is pre-trained using RetroMAE [10] and trained on large-scale pair data using contrastive learning [11]. The second scheme uses TF-IDF embedding in conjunction with the BGE embeddings (hybrid embedding). TF-IDF is a statistical algorithm that measures term importance by multiplying the term frequency with the inverse document frequency. Such an algorithm works well for surfacing terms best for document content identification (i.e. the terms that have high term frequency but low overall collection frequency) [8].

More formally, let $M_B \in \mathbb{R}^{m \times n}$ be an $n-$dimensional embedding matrix generated from BGE for m documents; BGE vectors are 1,024 dimensions, thus $n = 1,024$. Further, let $M_T \in \mathbb{R}^{m \times k}$ be an $k-$dimensional vector generated by TF-IDF, where $k >> n$. Because embedding vectors produced by TF-IDF may have more dimensions than their BGE counterparts, we used singular value decomposition (SVD [5]) to constrain the TF-IDF vectors to be of the same size as those produced by BGE as depicted by Eq.(1):

$$M_T \xrightarrow{SVD} M_T^{'} \tag{1}$$

where

$$M_T^{'} = U_p \Sigma_p V_p^T \tag{2}$$

Here, U, Σ, and V^T are the resulting component matrices from SVD obtained from M_T and constrained to the first p columns, where $p = \min(k, n)$. With this constraint in place, the hybrid embedding matrix, $M_E \in \mathbb{R}^{m \times n}$, is produced by combining the normalized rows of the BGE and SVD-constrained TF-IDF embeddings matrices as described below:

$$M_E[i] = \alpha \frac{M_B[i]}{||M_B[i]||} + (1-\alpha)\frac{M_T^{'}[i]}{||M_T^{'}[i]||}, \forall i \in \{1..m\} \tag{3}$$

An embedding weight, α, is chosen to emphasize either M_B or $M_T^{'}$. For the work described in this paper, we set $\alpha = 0.50$ to represent the average of M_B and $M_T^{'}$. In general, α can be adjusted to optimize the results.

In this hybrid embedding, TF-IDF captures term frequency and importance by analyzing the exact matching of terms and their

distribution across the documents while BGE hones in on deep semantic meaning and contextual information, understanding the nuance and relationship beyond exact term matches. By combining the embeddings, we simultaneously leverage precise term-level information from TF-IDF and rich contextual understanding from BGE, potentially resulting in a robust and comprehensive text representation. Note that, although we select BGE as a representative dense embedding model in this work, in practice, the hybrid embedding in Eq.(3) can be constructed using the output of any dense contextual embedding model for e_B.

3.2 Vector Database

The main function of the vector database is to store either dense embeddings or hybrid embeddings depending on the retrieval method, and to efficiently search for stored embeddings that are most similar to that of a query. In addition to the embeddings themselves, the vector databases also store relevant metadata, such as file name, node ID, and the document text chunks associated with these embeddings.

3.3 Generator

The generator component is powered by an LLM, in this study the Mixtral 8x7B instruct model [4]. The model excels in generating high-quality, fluent text that closely mimics human writing styles. This is essential for creating RFP responses that are correct in content and professional in tone. Moreover, due to its instruction-based training, the model is particularly effective at understanding the context provided by the retriever and processing it according to the provided instructions [4]. This ensures that the generated responses are relevant and accurately aligned with the RFP.

The model receives context relevant from the retriever, which includes the top five documents selected based on their relevance to the query. The LLM processes these inputs to synthesize and generate a coherent response. To enhance the model's understanding of the context and its role, system prompts and context prompts are given to the model to constrain its output.

3.4 Document Page Finder

The Document Page Finder improves the response provenance and ensures a human can quickly verify the generated responses. When a text chunk is retrieved and used by the generator to produce a response, it may seem trivial to determine which page of the source document the chunk comes from by comparing the dense embedding of the chunk with that of each document page under the assumption that the embedding captures the contextual information of the text. However, this approach is often inaccurate because a topic may span many pages in a document. Moreover, recall that the PDFs are converted to text files during chunking, causing a change in formatting and loss of metadata (filenames, page numbers, etc.), thereby rendering a simple reverse search impossible.

Thus, this study proposes to use TF-IDF embedding for reverse searching of page numbers (Algorithm 1). First, PDF documents are segmented by page and stored in a PDF database. Upon user query input, the Document Page Finder uses the metadata (filename) associated with the relevant context documents to subset the PDF database and select the candidate pages. The TF-IDF vectorizer is fitted on, and applied to, these candidate pages (line 6). The same TF-IDF vectorizer is applied to each relevant document used as context for a response, and cosine similarity between the context document and PDF pages is computed to rank page similarity (lines 7 to 9). The page with the highest similarity score is returned. This process ensures that the corpus of relevant context documents has the same vocabulary as the candidate documents.

Algorithm 1 Document Page Finder

Require: $pdf\_db \leftarrow$ set of all PDF documents chunked by page
1: **procedure** DocPageFinder($chunks$) ▷ Set of relevant chunks used to generate response
2: $sim\_score\_dict \leftarrow \{\}$ ▷ Dictionary where each element has a score and page number
3: **for** each c in $chunks$ **do**
4: **if** c in pdf_db **then**
5: $pdf\_subset \leftarrow$ SELECT(pdf_db, $c.filename$)
6: $tfidf\_pages \leftarrow$ TF-IDF($pdf\_subset.text$)
7: $tfidf\_query \leftarrow$ TF-IDF($query$)
8: **for** each $page$ in $tfidf\_pages$ **do**
9: $sim\_score \leftarrow$ COSINE($tfidf\_query$, $page$)
10: $sim\_score\_dict[sim\_score] \leftarrow page.page\_num$
11: **end for**
12: **end if**
13: **end for**
14: **return** $page\_num$ from sim_score_dict where sim_score is maximum
15: **end procedure**

If the document is originally a Microsoft Word file, it is converted into PDF and follows Algorithm 1. If the document is originally a CSV, the retriever will return the name of the CSV as well as which sheet it was from.

4 EXPERIMENTAL SETUP

4.1 Dataset

4.1.1 Documents. The dataset of relevant documents to be retrieved by a RAG system consists of 147 documents of three different types: PDF (84 documents), CSV (61), and Microsoft DOCX (2). These documents are curated from answers (human-generated) to past RFPs and, therefore, can enhance the model's ability to generate content that adheres to pertinent legal, ethical, and technological standards. These archival RFPs are instrumental in teaching the model the structure and tone typically favored in successful proposals; the system learns to identify key phrases and topics that are crucial for a compelling response. The 147 documents are converted to a textual representation from their native document representation to aid in *chunking*, a process of dividing the text corpus into smaller, more manageable segments to enable the RAG system to process and understand large corpora. In all, 2,212 chunks are obtained from the documents by using an overlapped chunking strategy, i.e., each chunk includes some context from the previous and following chunks. Each chunk has an overlap of 20 tokens [2]. BGE enforces a token limit of 512 tokens, thus all of the document

[2]https://docs.llamaindex.ai/en/stable/optimizing/basic_strategies/basic_strategies/

chunks consisted of at most 500 tokens, with the constraint that no chunk is truncated by the embedding model.

4.1.2 Test sets. Two RFPs are kept aside to evaluate and compare the performance of the BGE embeddings with the embeddings produced by the hybrid model. These test sets are previously completed RFPs from two different clients. Both RFPs include a mix of compliance, security, policy, and general questions about operations. The RFPs from Client A and Client B consist of 104 questions and 371 questions, respectively. The questions from Client A are mostly answered with a simple yes/no, while those from Client B are free-response questions. Thus, the correctness of the response generated by the system is manually verified by comparing the machine response against the provided ground truth.

To evaluate the Document Page Finder, a separate manually completed RFP questionnaire consisting of 13 questions is utilized.

4.2 Models

The following embedding and LLM models are used in this work:

- Dense Retrieval:
  - Retriever embedding model: bge-large-en-v1.5
  - Generator LLM: mixtral-8x7b-instruct-v0.1.Q5_K_M.gguf
- Hybrid Retrieval:
  - Retriever embedding models: bge-large-en-v1.5 for M_B in Eq.(3), and TF-IDF for $M_T^{'}$ in Eq.(1).
  - Generator LLM: mixtral-8x7b-instruct-v0.1.Q5_K_M.gguf

where bge-large-en-v1.5 is a BGE [11] dense contextual embedding model, and mixtral-8x7b-instruct-v0.1.Q5_K_M.gguf [3] is a quantized Mixtral 8x7b Instruct model [4] for memory and computation efficiency.

TF-IDF is also used by Document Page Finder to search the exact location of a retrieved text chunk in the source document.

4.3 RAG System

Due to the proprietary nature of RFP documents and answers, the entire RAG system, based on LlamaIndex[4], is deployed along with all embedding and large language models inside the enterprise network.

The system is initialized either with dense retrieval or hybrid retrieval method. Chunking is performed first for all documents as described in Section 4.1.1, then for each chunk either a dense embedding or a hybrid embedding is generated according to the initialization parameter, and stored in the vector database.

At inference time, for each RFP question, the retriever generates an embedding corresponding to the retrieval method of the system, then searches for the top 5 most relevant context chunks in the vector database based on cosine similarities of the embeddings. Subsequently, the generator incorporates these 5 context chunks into the LLM input prompt for response generation. Finally, before the response is returned to the user, the Document Page Finder is invoked for each context chunk to find the corresponding page number in the source document. This procedure allows the system to return not only a suggested response but also links to the specific pages of relevant source documents for manual verification.

[3]https://huggingface.co/TheBloke/Mixtral-8x7B-Instruct-v0.1-GGUF
[4]https://docs.llamaindex.ai/en/stable/api_reference/chat_engines/condense_plus_context/

5 RESULTS

5.1 Dense Retrieval

As described in Section 4.1.2, two client questionnaires are used as test sets for evaluation. The ground truths for Client A are binary Yes or No, whereas answers for Client B are freeform text.

Table 1 shows the results of applying dense retrieval to Client A dataset, with an overall accuracy of 85.58%. With 5 context chunks retrieved based on their BGE embedding similarity to that of the RFP question, the LLM indicates there is no relevant information that can be used to answer 4 out of 104 questions in Client A set. For the remaining 100 questions, the system achieves a balanced accuracy of 67.51%, a precision of 91.30%, and a recall of 96.55% (We report balanced accuracy due to the class-imbalanced dataset.). However, the system seems to be biased towards giving "yes" answers: out of the 89 questions where the ground truth is "yes," the model is able to predict correctly 94.38% of the time, while only being correct 33.33% of the time when the ground truth is "no." This appears to be the result of the nuances in human interaction in the training data for the LLM. Yin et al. [12] discussed how LLMs reflect human communication traits and cultural norms, suggesting that polite language in prompts often garners more compliant and effective responses. It is possible that the polite language in RFP questions biases the LLM in responding with an affirmative answer.

Ground Truth	RAG Response			Total
	Yes	No	No Context	
Yes	84	3	2	89
No	8	5	2	15
Total	92	8	4	104

Table 1: Dense Retrieval for Client A

For Client B, the system achieves an accuracy of 94.07% as shown in Table 2. The system is not able to find relevant context to provide answers for 2 out of 371 questions, yielding a no-context rate of 0.53%.

	RAG Response			Total
	Correct	Incorrect	No Context	
Total	349	20	2	371

Table 2: Dense Retrieval for Client B

5.2 Hybrid Retrieval

For hybrid retrieval, the RAG vector database is reconstructed with the hybrid embeddings (Eq.3) of the same document chunks as those used in dense retrieval. Moreover, each RFP question entering the RAG system is also embedded using Eq.3).

The hybrid retrieval performance for Client A is presented in Table 3. Compared to dense retrieval (Table 1), the overall accuracy of hybrid retrieval increases from 85.58% to 86.54% while the no-context rate is reduced from 3.85% to 0.96%, resulting in a precision

of 89.47% and recall of 96.59%. The balanced accuracy is 64.96%. The slight decrease in precision and balanced accuracy can be attributed to the additional incorrect answers in the process of eliminating no-context cases.

Similar improvement is seen in Table 4 for Client B, where the hybrid retrieval achieves an accuracy of 95.15% with relevant context found for all RFP questions in the test set.

Ground Truth	RAG Response			Total
	Yes	No	No Context	
Yes	85	3	1	89
No	10	5	0	15
Total	95	8	1	104

Table 3: Hybrid Retrieval for Client A

	RAG Response			Total
	Correct	Incorrect	No Context	
Total	353	18	0	371

Table 4: Hybrid Retrieval for Client B

Table 5 compares the results of hybrid retrieval with those of dense retrieval; even though the improvement in overall accuracy is marginal, the reduction in no-context responses is significant: since each generated response needs to be reviewed by humans due to the rigorous requirement of RFP, providing links to relevant document sections are crucial to the process efficiency as well as verification accuracy. Source document attribution not only helps reduce the time it takes to manually search for answers for no-context questions among all source documents in the database, but also lowers the possibility of overlooking pertinent information by human reviewers. Therefore, optimizing for reducing no-context cases via hybrid retrieval is the preferred solution for RFP response generation, even though it may potentially result in slightly lower precision.

Client	Metric (%)	Dense	Hybrid
A	Accuracy	85.58	**86.54**
	Precision	**91.30**	89.47
	Recall	96.55	**96.59**
	No-Context Rate	3.85	**0.96**
B	Accuracy	94.07	**95.15**
	No-Context Rate	0.53	**0**

Table 5: Metrics Comparison for Client A and B

Using Client A's RFP, the hybrid retriever was timed and compared to the inference time of the LLM to measure scalability. For the 104 questions, the retrieval stage latency had a mean of 0.1053s and a standard deviation of 0.0045s. The inference stage latency had a mean of 7.9173s and a standard deviation of 3.0022s. Comparing these statistics, the retrieval stage latency is much lower compared to the time it takes for the LLM to digest the retrieved documents and generate a response. The low standard deviation shows that the retrieval stage latency is relatively stable.

5.3 Discussion

To illustrate the differences between dense retrieval and hybrid retrieval, a sample RFP question along with the top relevant chunks identified using either approach is analyzed in Example 5.1. The precise answer to the question in the relevant chunk is highlighted.

Example 5.1. **Question**: If the solution stores data, is it encrypted with AES 256? If no, explain.

Top 3 Chunks from Dense Retrieval (Excerpts):

(1) Authentication factors (passwords, passphrases, or cryptographic keys) that allow access to unencrypted data are stored securely • 3.6.1) Procedures are defined and implemented to protect cryptographic keys used to protect stored account data against disclosure and misuse...
(2) Is [redacted] data encrypted in storage? Acceptable options are logical encryption or disk encryption where access is managed separately and independently of native operating system controls. DS-8 C 33.1 If a database is in use, is column/field level encryption implemented to protect sensitive [redacted] data (such as SSN, credit card numbers, etc.)? If response is "no", please confirm that no database is in use. DS-8.1 C...
(3) Indicate whether disk encryption is used. (yes/no) [redacted] If "yes," complete the remainder of 3.4.1.a, 3.4.1.b, and 3.4.1.c. If "no," mark the remainder of 3.4.1.a, 3.4.1.b and 3.4.1.c as "Not Applicable.' Describe the disk encryption mechanism(s) in use. [redacted]. For each disk encryption mechanism in use, describe how the configuration verified that logical access to encrypted file systems is separate from the native operating system's authentication mechanism. [redacted]. For each disk encryption mechanism in use...

Top 3 Chunks from Hybrid Retrieval (Excerpts)

(1) Audio recorded phone calls are either automatically redacted of sensitive cardholder data elements, are encrypted with client-provided public encryption keys (for which [redacted] personnel do not have access to the associated private keys), or are stored encrypted with [redacted]-managed asymmetric key pairs for limited durations before being deleted at per-client defined retention periods...
(2) Key-encrypting keys are stored separately from data-encrypting keys. Describe how system configurations and key storage locations verified that, wherever key-encrypting keys are used: • **Key-encrypting keys are at least as strong as the data-encrypting keys they protect. SQL Database DEK: AES-256 SQL Database KEK: [redacted] SQL Database Master Key (DMK): AES-256 Call Recording DEK: [redacted] Call Recording KEK: AES-256**...
(3) Departures from this policy will be permitted only if approved in advance and in writing by the Information Security Officer.
3. Policy

> It is the policy of [redacted] that all sensitive data will be encrypted while at rest and during transit across public networks to protect it from internal and external threats...

In Example 5.1, both dense and hybrid retrieval approaches find chunks of similar topics, but the hybrid retrieval is able to pinpoint a chunk containing the exact phrase that is present in the RFP question. While the dense BGE embedding captures deep semantic meanings and contextual information, understanding nuances and relationships beyond exact term matches, the sparse TF-IDF embedding captures term importance and focuses on the exact matching of terms (e.g., AES 256) and their distribution across documents. By combining the two embeddings, a more comprehensive text representation is created, leveraging both precise term-level information (from TF-IDF) and rich contextual understanding (from BGE). This combination results in the correct document being surfaced to answer the question.

5.4 Document Page Finder

An existing manually completed RFP questionnaire consisting of 13 questions is used to evaluate the performance of the novel Document Page Finder. Each question is submitted to the RFP completion system. For a specific question, the page number suggested by the Document Page Finder is verified manually for each retrieved document (these are the original PDF documents).

Although the dense contextual embedding model has already generated embeddings for each chunk by this point, they are found to be less robust for mapping the retrieved text chunks to their corresponding PDF pages. Theoretically, the TF-IDF algorithm excels in identifying the best terms for distinguishing individual documents relative to the entire corpus. This characteristic renders TF-IDF particularly effective for locating the corresponding page for a specific document chunk [7]. In contrast, BGE embeddings are more adept at capturing and matching semantic meaning, making them less useful for the Document Page Finder.

Indeed, experiments show that the accuracy of using BGE embedding to find source document page numbers is lower compared to TF-IDF embedding. Out of the 13 questions, a total of 24 retrieved documents are PDFs and thus are supported by the Document Page Finder. The system using TF-IDF achieves 100% accuracy as shown in Table 6.

Retrieved PDFs	Correct Matches	
	BGE	TF-IDF
24	21	24

Table 6: Document Page Finder Performance

6 CONCLUSION AND FUTURE WORK

A comparative analysis of documents retrieved through dense and hybrid retrievals reveals that both approaches yield content on highly similar and relevant topics. The hybrid retriever, employing a weighted combination of embeddings, integrates the deep contextual understanding of the BGE embedding [11] with the ability of the TF-IDF embedding to identify unique and significant terms. This combination enhances the efficacy of the hybrid embedding in retrieving documents that encompass critical terms relevant to the query, thereby reducing no-context responses for RFP questions. Additionally, the high-performing Document Page Finder enhances provenance by providing detailed references for each generated answer. Moreover, the scalability of this system is not hindered by the hybrid retriever, and the increase in retrieval time should be minuscule compared to the time it takes for the LLM to make an inference when the number of documents increase. These improvements render the RAG architecture more efficient and verifiable, making it suitable for any enterprise application that demands high accuracy and accountability.

In this study, the chunk size is selected to align with the token limit imposed by the BGE embedding model. Given that the generation of document embeddings is a critical factor for the hybrid retriever, it can be beneficial to experiment with various chunk sizes. Such experimentation may capture varying amounts of context within a single vector, thereby facilitating the determination of the optimal chunk size for achieving the most relevant retrieval results. Another possible area of optimization is the α parameter shown in Eq.(3). This study uses 0.5 to represent the average of the BGE embedding and the TF-IDF embedding, but this value can be optimized according to the use case to find the optimal combination of dense contextual and sparse statistical embeddings. By addressing these nuances, future work will significantly contribute to refining the robustness, impartiality, and accuracy of RAG systems, thereby enhancing their reliability and applicability in diverse contexts.

REFERENCES

[1] Wenqi Fan, Yujuan Ding, et al. 2024. A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery Data Mining*.

[2] Kelvin Guu, Kenton Lee, et al. 2020. Retrieval Augmented Language Model Pre-Training. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 3929–3938.

[3] Amir Jalilifard, Vinicius Fernandes Caridá, et al. 2021. Semantic Sensitive TF-IDF to Determine Word Relevance in Documents. In *Advances in Computing and Network Communications*. Springer Singapore, Singapore, 327–337.

[4] Albert Q. Jiang, Alexandre Sablayrolles, et al. 2024. Mixtral of Experts. arXiv:2401.04088 [cs.LG] https://arxiv.org/abs/2401.04088

[5] V. Klema and A. Laub. 1980. The singular value decomposition: Its computation and some applications. *IEEE Trans. Automat. Control* 25, 2 (1980), 164–176. https://doi.org/10.1109/TAC.1980.1102314

[6] Patrick Lewis, Ethan Perez, et al. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 9459–9474.

[7] Juan Enrique Ramos. 2003. Using TF-IDF to Determine Word Relevance in Document Queries. https://api.semanticscholar.org/CorpusID:14638345

[8] Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing Management* 24, 5 (1988), 513–523.

[9] Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation* 28, 1 (1972), 11–21.

[10] Shitao Xiao, Zheng Liu, et al. 2022. RetroMAE: Pre-Training Retrieval-oriented Language Models Via Masked Auto-Encoder. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 538–548.

[11] Shitao Xiao, Zheng Liu, et al. 2024. C-Pack: Packed Resources For General Chinese Embeddings. In *Proc. of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24)*. 641–649.

[12] Ziqi Yin, Hao Wang, et al. 2024. Should We Respect LLMs? A Cross-Lingual Study on the Influence of Prompt Politeness on LLM Performance. arXiv:2402.14531 [cs.CL] https://arxiv.org/abs/2402.14531