Task 2 - Optimising RAG: Two Innovative Techniques for Optimising the RAG Model

Technique 1: Hybrid Retrieval with Sparse and Dense Embeddings

Traditional RAG models rely solely on dense embeddings (e.g., from Sentence Transformers or OpenAI embeddings) for retrieving relevant documents. However, dense embeddings may not always capture keyword-based relevance, especially for domain-specific or rare terms. A hybrid retrieval approach combines **sparse embeddings** (e.g., BM25 or TF-IDF) with **dense embeddings** to improve retrieval accuracy.

How it works:

1. **Sparse Retrieval**: Use a sparse retrieval method like BM25 to retrieve documents based on keyword matching. This ensures that documents containing exact query terms are prioritized.
2. **Dense Retrieval**: Use dense embeddings (e.g., Sentence Transformers or OpenAI embeddings) to capture semantic similarity between the query and documents.
3. **Fusion**: Combine the results from sparse and dense retrieval using a fusion method like reciprocal rank fusion (RRF) or weighted scoring. This ensures that both keyword relevance and semantic relevance are considered.

Benefits:

- Improved retrieval accuracy for both keyword-based and semantic queries.
- Better handling of rare or domain-specific terms that may not be well-represented in dense embeddings.
- Enhanced robustness in scenarios where semantic similarity alone may not suffice.

Implementation:

- Use libraries like rank_bm25 for sparse retrieval and integrate it with the existing dense retrieval pipeline.
- Implement a fusion mechanism to combine results from both retrieval methods.

Technique 2: Query Expansion and Rewriting

Query expansion and rewriting aim to improve retrieval performance by enhancing the query's representation. This technique is particularly useful when the user's query is ambiguous, too short, or lacks context.

How it works:

1. **Query Expansion**: Expand the query by adding synonyms, related terms, or contextually relevant phrases. For example, if the query is "How to fix XYZ3000?", the expanded query could include terms like "troubleshoot," "repair," or "device not working."
2. **Query Rewriting**: Rewrite the query to make it more specific or align it with the document structure. For example, rewrite "How to set up XYZ3000?" to "Steps to set up XYZ3000 device."
3. **LLM-Based Rewriting**: Use a large language model (LLM) like GPT-4 to generate multiple versions of the query, capturing different aspects of the user's intent.

Benefits:

- Improved retrieval of relevant documents by addressing query ambiguity or lack of context.
- Better alignment between the query and the document structure, leading to more accurate results.
- Enhanced user experience by providing more relevant answers.

Implementation:

- Use libraries like spaCy or nltk for synonym extraction and query expansion.
- Integrate an LLM (e.g., GPT-4) for query rewriting, generating multiple query variants.
- Combine the expanded/rewritten queries with the retrieval pipeline to retrieve a broader set of relevant documents.

By implementing these techniques, the RAG model can achieve higher retrieval accuracy, better alignment with user intent, and improved overall performance.