Literature review

Deng Kaina dengkn@mail2.sysu.edu.cn 2025/9/22

1. Sparse Retrieval

- **What are the authors trying to do?**
  To create a relevance-based calculation method for retrieval weights in order to improve document ranking. (The paper assumes a document's relevance score is the sum of the weights of its terms).
- **How was it done prior to their work, and what were the limits of current practice?**
  - **Prior methods:** Used term frequency or Boolean retrieval models.
    - **Boolean model:** Only considers if a term is present (0/1) and cannot reflect the differing importance of various terms.
    - **TF model:** Simply uses term frequency as a weight but does not account for the distinction between relevant and non-relevant documents.
  - **Limitations:**
    - They did not directly utilize information from "**relevance feedback**.", which is the innovtion of this paper. They could not dynamically adjust term weights to optimize ranking. For all queries, a term would have the same weight across different retrieval requests. However, when considering relevant documents, the same term might have different importance for different requests.
- **What is new in their approach, and why do they think it will be successful?**
  - **Innovation:** They proposed a weighting formula based on the statistics of relevant and non-relevant documents (the precursor to the later BM series of formulas).
  - The weight function depends on:
    - **N**: The total number of documents in the collection.
    - **R**: The number of relevant documents for the current query *q*.
    - **n**: The number of documents containing a specific term *t*.
    - **r**: The number of documents that are both relevant and contain the term *t*.
  - It is worth noting that when estimating R and r using user relevance feedback, a **corrected estimate (+0.5 bias)** is needed.

$$w_t \approx \log \frac{(r+0.5)/(R-r+0.5)}{(n-r+0.5)/(N-n-R+r+0.5)} (4)$$

  - For a query *Q* (containing several query terms q_i) and a document *D*, the BM25 score is typically written as:

$$\text{score}(D, Q) = \sum_{q_i \in Q} \text{IDF}(q_i) \cdot \frac{f(q_i, D)\,(k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

Where:
* $f(q_i, D)$: The term frequency of the query term q_i in document D.
* $|D|$: The length of the document (usually in number of words).
* avgdl: The average document length in the corpus.
* k_1 and b: Tunable hyperparameters that control the intensity of **term frequency saturation** and **length normalization**, respectively. k_1 should not be linear because the contribution to relevance from a term appearing 10 times versus 20 times in a document has diminishing returns. b is used to penalize the model for retrieving overly long documents, as they are more likely to be retrieved by chance.
* $\text{IDF}(q_i)$: **Inverse Document Frequency, often using the smoothed Robertson–Spärck Jones (RSJ) form, which is Formula (4) in the paper**.

- **What are the mid-term and final "exams" to check for success?** (i.e., How is the method evaluated?)
    i. The authors compared the Recall-Precision Curves for formulas F1-F4. F4 performed the best.
    ii. Miller's experiment used the F1 weighting formula proposed by the authors and applied it to a real, large-scale MEDLARS medical literature database, comparing the precision and recall of two search methods.

2. Dense Retrieval

- **What are the authors trying to do?** Articulate their objectives.
  The goal of Karpukhin et al. in DPR is to use learned dense vector representations (dense embeddings) to replace traditional sparse bag-of-words models (like BM25). This is done to capture semantic meaning, synonyms, and semantic matching, thereby retrieving more relevant passages in open-domain QA / passage retrieval tasks and improving the accuracy of the downstream question-answering model.
- **How was it done prior to their work, and what were the limits of current practice?**
    - **Previous IR methods:**
        - **Sparse:** BM25.
        - **Dense:** ORQA (Lee et al., 2019) proposed a complex Inverse Cloze Task (ICT) objective, which predicts a block of text containing a masked-out sentence. ORQA outperformed BM25.
        - Used MIPS algorithms for efficient vector retrieval.
    - **Limitations:**

    a. ICT pre-training is computationally intensive.
    b. Because the context encoder is not fine-tuned using pairs of questions and answers, the corresponding representations could be suboptimal.
- **What is new in their approach, and why do they think it will be successful?**
  - With an appropriate training setup, simply fine-tuning on existing question-paragraph pairs can significantly surpass BM25. Additional pre-training may not be necessary.
  - After training, retrieval can be done using a vector index (like FAISS).
  - They verified that higher precision in IR can improve the accuracy of QA.
  - They used a negative sampling technique called **In-batch negatives**, which cleverly uses other passages within the same batch as negative examples for the current question. These negative examples are often topically similar to the positive sample (as they may come from the same corpus), thus serving as hard negatives.
- **What are the mid-term and final "exams" to check for success?**
  - **Retrieval-level evaluation:** top-k retrieval accuracy.
  - **End-to-end QA evaluation:** The retriever is combined with a reader/generator model to evaluate the QA (Exact Match) Accuracy, which is then compared to systems like ORQA and REALM. This demonstrates the effectiveness of DPR being trained solely on question-answer pairs.

3. RAG

- **What are the authors trying to do?**
  To combine a retriever and a generator, dynamically retrieving relevant documents to provide additional context. This paper presents the model architecture and training method for RAG. (LangChain is an engineering implementation of RAG).
- **How was it done prior to their work, and what were the limits of current practice?**
  i. Generative models relied solely on model parameters and were prone to hallucination.
  ii. Systems relying on a retriever+reader combination were limited, as the reader could only generate text spans and lacked generalization capability.
- **What is new in their approach, and why do they think it will be successful?**
  - This paper combines a retriever and a generator, providing a non-parametric method for improving LLMs.
  - There are two paradigms: RAG-Sequence and RAG-Token.
- **What are the mid-term and final "exams" to check for success?**
  - Open-domain QA, Jeopardy Question Generation, and Fact Verification.
  - Evaluation is performed using Exact Match (EM) scores.

Concluding Section

- **Who cares? What difference does the author's results make?**
  - **Who cares?** People working on retrieval and question-answering.
  - **Relevance Weighting of Search Terms:** Introduced statistical techniques that use relevance information to assign weights to search terms, thereby improving retrieval performance.
  - **DPR:** By using a dual-encoder architecture, DPR encodes questions and text passages into the same high-dimensional semantic space, determining relevance by calculating vector distance (e.g., dot product).
  - **RAG:** Combines a retriever and a generator, providing the retrieved context to an LLM as the basis for generating an answer, which reduces model hallucination.
- **What are the risks?** What are the potential failure modes or downsides of these approaches?
  - **Relevance Weighting (BM25):** It relies entirely on lexical, literal matching. If the user's query and the documents in the knowledge base use different words to describe the same concept (synonyms, paraphrases), BM25 may fail to find the relevant documents.
  - **DPR:**
    - Training a DPR model requires significant GPU resources and high-quality labeled data;
    - it suffers from domain generalization problems and needs retraining for new domains;
    - it can be less sensitive to keywords.
  - **RAG:** The retriever+generator paradigm itself has few downsides. However:[1]
    - Current retrievers are mainly text-based. How to retrieve from **tabular data**[2] and how to convert and store **multi-modal documents** as vectors are open questions.
    - The essence of RAG is search, and it works on the premise that the answer can be "found" based on the user's query. However, in many cases, this premise does not hold, such as with vague questions with unclear intent, or so-called "multi-hop" questions that require synthesizing an answer from multiple sub-problems. The answers to these questions cannot be obtained by simply searching the query, as there is a clear **semantic gap** between the question and the answer. (Current research includes GraphRAG[3], which uses LLMs to generate associative information across chunks).
- **Synthesis:** Briefly explain how these three technologies fit together. How do sparse and dense retrieval support the RAG framework? What are the pros and cons of using one retrieval method over the other in a RAG system?
  - They have a progressive relationship. BM25 and DPR are both retrievers, with the difference being that one uses statistical information while the other uses dense vectors. RAG is composed of a retriever and a generator.
  - Compared to a sparse retriever, a dense retriever has the following pros and cons:
    - **Pro:** Captures semantic connections (even if two words are not the same), achieving better performance than BM25.

        - **Con:** Incurs training costs.
    - In real-world applications, we can use **hybrid retrieval**[4], which combines the BM25 score, the DPR score, and even summaries of entities/relations/communities from a knowledge graph in some way to produce the final ranking, reducing the semantic gap.

1. https://blog.csdn.net/sinat_39620217/article/details/147386100 ↩
2. https://arxiv.org/abs/2410.04739v1 ↩
3. https://arxiv.org/abs/2404.16130 ↩
4. https://arxiv.org/html/2408.04948v1 ↩