Retrieval-Augmented Generation for Natural Language Processing: A Survey

Shangyu Wu
City University of Hong Kong,
MBZUAI

Ying Xiong
MBZUAI

Yufei Cui

Haolun Wu
McGill University, Mila

Can Chen
McGill University, Mila

Ye Yuan
McGill University, Mila

Lianming Huang
City University of Hong Kong

Xue Liu
McGill University, Mila

Tei-Wei Kuo
National Taiwan University

Nan Guan
City University of Hong Kong

Chun Jason Xue
MBZUAI

ABSTRACT

Large language models (LLMs) have demonstrated great success in various fields, benefiting from their huge amount of parameters that store knowledge. However, LLMs still suffer from several key issues, such as hallucination problems, knowledge update issues, and lacking domain-specific expertise. The appearance of retrieval-augmented generation (RAG), which leverages an external knowledge database to augment LLMs, makes up those drawbacks of LLMs. This paper reviews all significant techniques of RAG, especially in the retriever and the retrieval fusions. Besides, tutorial codes are provided for implementing the representative techniques in RAG. This paper further discusses the RAG training, including RAG with/without datastore update. Then, we introduce the application of RAG in representative natural language processing tasks and industrial scenarios. Finally, this paper discusses the future directions and challenges of RAG for promoting its development.

1 INTRODUCTION

Large language models (LLMs) [71, 108, 114, 138, 164] have achieved significant advancements in recent years and have become the cornerstone of various applications in the field of natural language processing (NLP). These LLMs are typically pre-trained on a large amount of natural language corpus and then fine-tuned on the specific downstream tasks' datasets. Recent works [3, 53, 106, 117] demonstrate the success of LLMs can be explained by the fact that language models act as knowledge bases, which refers to implicitly storing the knowledge learned from training datasets in the parameters as internal memory and generating responses by retrieving answers from memory. To store more knowledge for better generation performance, existing works generally enlarge the memory capacity by increasing the volume of parameters [1, 11, 54, 78].

Although existing LLMs have shown great power, there are still several challenges hindering the development of LLMs. One of the most prominent challenges is the hallucination problem [23, 69, 70], which refers to the tendency of LLMs to generate responses that are coherent and fluent but factually incorrect. Another big challenge is the knowledge update issue. To update the knowledge stored in the LLMs' internal memory [106, 146, 168], it is necessary to

retrain/fine-tune LLMs with new data, which is a costly process. Another challenge for general LLMs is lacking of domain-specific expertise [20, 133, 134, 166]. Training a domain-specific LLM, however, demands considerable manpower for dataset collection.

To address these challenges, recent works [10, 49, 87] have proposed leveraging an external knowledge database to augment LLMs, known as retrieval-augmented generation (RAG). By supplying LLMs with retrieved relevant factual information, the hallucination problem can be alleviated to some extent. Besides, the knowledge update issue can also be addressed by updating the external knowledge database, which can augment LLMs with up-to-date knowledge. RAG can also convert a general LLM into a domain-specific LLM by constructing and utilizing a domain-specific knowledge database. Therefore, RAG plays an important role in augmenting the functionality of LLMs, making them more accurate, knowledgeable, and reliable in a wide range of applications.

Contributions: This paper reviews all techniques involved in RAG for natural language processing. Although there are several survey papers for RAG [41, 59, 88, 162, 171], our survey still has some key insights,

(1) This paper systematically introduces each component of RAG, including details about the retriever from building to querying, and techniques of the retrieval fusions with tutorial codes.
(2) This paper exhibits different RAG training strategies, including RAG with/without datastore update.
(3) This paper further discusses the applications of RAG on downstream NLP tasks and practical NLP scenarios.
(4) This paper finally identifies promising future directions for exploring and main challenges for addressing.

The remainder of this paper is organized as follows. Section 2 gives an overview of RAG. Section 3 and Section 4 comprehensively introduce all technical details used in retrievers and retrieval fusions. Section 6 presents how to train the RAG with/without new knowledge. Section 7 presents the techniques used in representative NLP tasks. Section 8 shows the applications of RAG in practical NLP scenarios. Section 9 discusses the future directions of RAG. Section 10 makes a final conclusion of this paper.
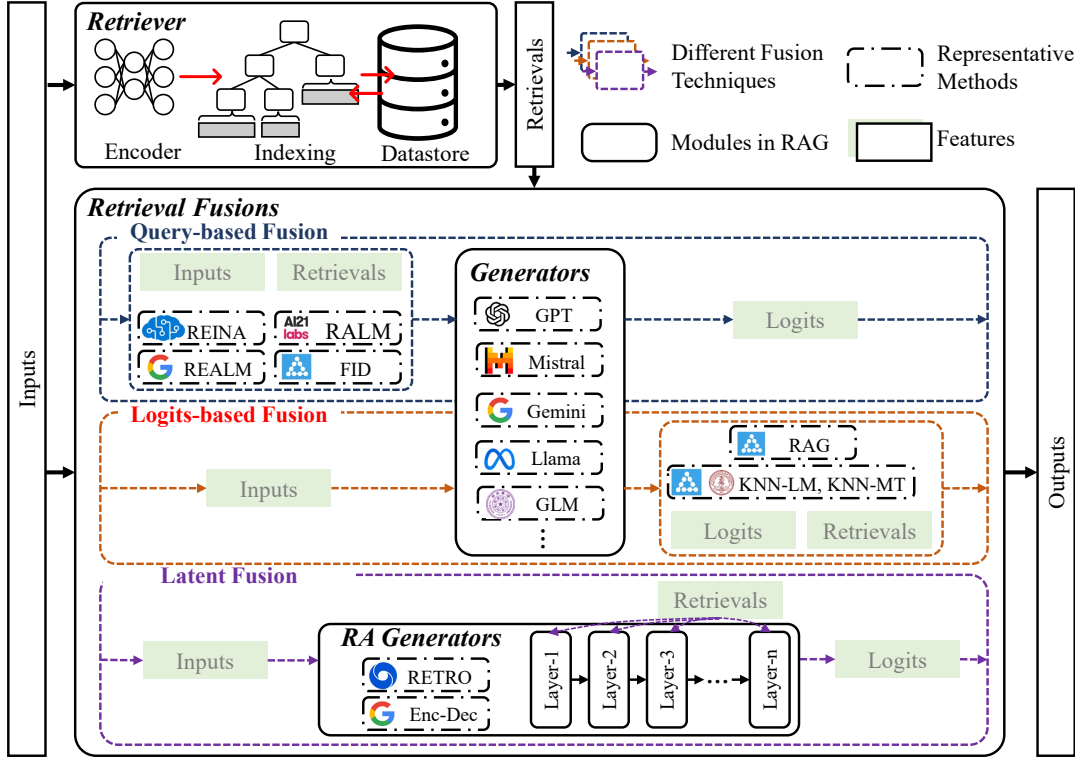
Figure 1: The overview of retrieval-augmented generation for natural language processing.

2 OVERVIEW OF RETRIEVAL-AUGMENTED GENERATION

This section gives an overview of RAG for NLP. As shown in Figure 1, RAG typically consists of three modules, the retriever, the generator, and retrieval fusions.

Retriever module usually comprises three components: an encoder for encoding inputs into embeddings, an efficient indexing that supports approximate nearest neighbor search, and the datastore for storing external knowledge in the form of key-value pairs. The main challenge in the retriever module is *finding the optimal trade-off between retrieval efficiency and retrieval quality*. The retrieval efficiency refers to how fast the relevant information can be obtained, which involves accelerating encoding, efficient indexing, batch querying in the datastore, etc. The retrieval quality refers to how relevant the information can be retrieved, which involves chunk representation learning, advanced approximate nearest neighbor search algorithms, etc.

Retrieval Fusions aims to leverage the retrieved information to augment the generation. These fusion techniques can be categorized into three major types: query-based fusion, latent fusion, and logits-based fusion. The query-based fusion augments inputs with retrievals before feeding them into the generators. The logits-based fusion focuses on the output logits of generators and fuses the retrievals logits for more robust logits. The latent fusion refers to introducing retrieval representations into the latent representations of generators, thus implicitly improving the models' performance.

Generator module can be classified into two branches of generators: default generators and retrieval-augmented (RA) generators. The default generators include most pre-trained/fine-tuned large language models, such as GPT-series models [11, 114, 118, 119], Mistral models [71], and Gemini-series models [4, 108, 124]. The RA generators refer to the pre-trained/fine-tuned generators that consist of modules for fusing retrievals, such RETRO [10] and Enc-Dec [93]. Those generators generate responses or make predictions.

The workflow of RAG involves three steps: (1) retrieving the relevant information from external databases based on given inputs; (2) fusing the retrieved information with inputs or intermediate states based on the fusion techniques; (3) making predictions by generators based on the input and corresponding retrievals.

3 RETRIEVER

Figure 2 shows the two stages for using the retriever, which involves first building the retriever and then querying the retriever. The following sections will introduce details about each stage.

3.1 Building the Retriever

This section will explain how to build a retriever using a large natural language corpus. As shown in Figure 2 (a), the process involves three steps: chunking corpus, encoding chunks, and building the vector database. Specifically, building the vector database includes building the ANN index and storing the data with key-value pairs.
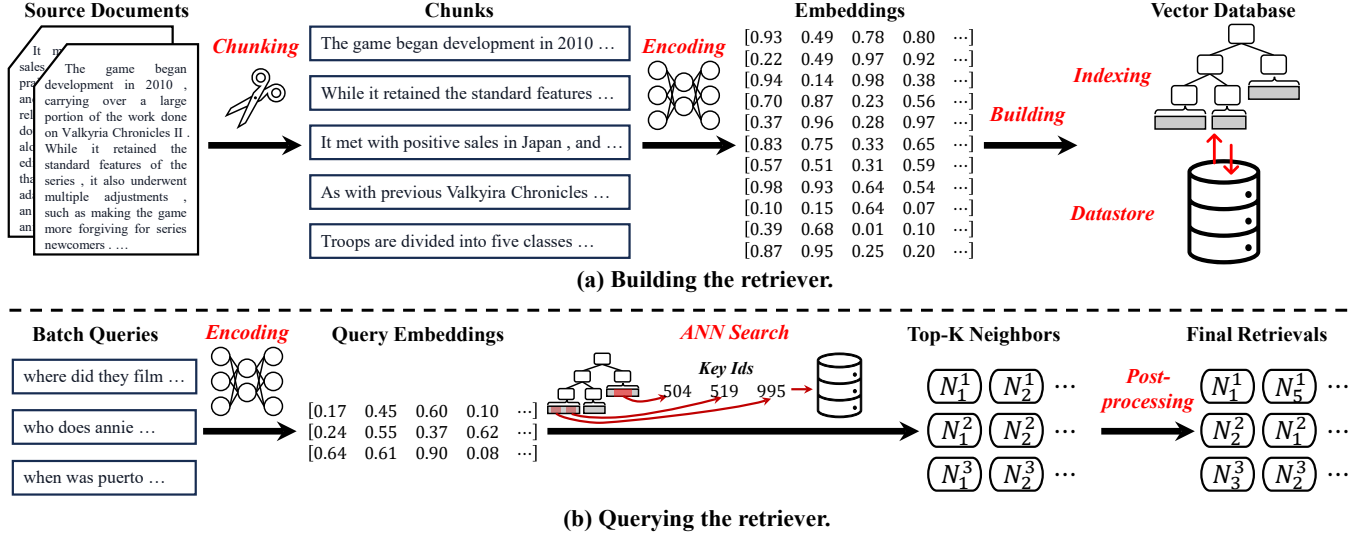
(a) Building the retriever.

(b) Querying the retriever.

Figure 2: Two stages of using the retriever.

3.1.1 Chunking Corpus. Chunking techniques generally refer to dividing large documents into small text chunks [10, 13, 43, 65, 111], which is an indispensable key step in the process of building the retriever. The intuitions behind chunking techniques are, (1) The texts or embeddings used for the indexing should be semantically independent, containing one core idea for models to encode. Short texts are more likely to be ambiguous, for example, the word "apple" can refer to a fruit or a company. (2) Encoding a long sequence document would result in considerable resource overheads when using existing transformer-based models, while processing shorter text chunks can significantly accelerate the encoding process and save memory costs. Therefore, the main challenge of the chunking techniques is to find the best chunking size to make a better trade-off between text semantics and encoding efficiency.

To solve the above challenge, there are three key points that need to be considered when determining the chunking size:

(1) **Task's property.** Different tasks may benefit from different kinds of retrieval chunks. For example, question-answer tasks may prefer short phrases, while summarization tasks may prefer long documents.

(2) **Encoder's property.** Different encoder models have varying encoding capabilities on texts with different lengths. For example, models in the sentence-transformer [125] behave better on a single sentence, while the text-embedding-ada-002 [113] is good at longer texts.

(3) **Query's property.** The length of the user's queries should be aligned with the chunking size, which implicitly aligns the amount of contextual information in chunks with that in queries, thus improving the relevance between queries and retrievals. For example, a retrieval database built on short phrases may be useless for queries with long documents.

Overall, there is no golden rule for determining the chunking size, and it depends on the specific RAG scenarios.

There are basically three types of chunking techniques, including **the chunking with fixed length**, **the semantic chunking**,

and **the content-based chunking**. Chunking with fixed length is the simplest way to split documents sequentially using a length hyperparameter. The semantic chunking cuts documents based on semantics, such as the period character or the newline character that represents the end of the sentence. Existing state-of-the-art natural language processing toolkits, such as NLTK [112] and spaCy [33], have provided convenient sentence-cutting methods. The content-based chunking segments documents according to the unique structural characteristics. For example, electronic medical records can be easily segmented based on the sections, or programming codes can be segmented based on function blocks.

3.1.2 Encoding Chunks. Encoding refers to numericalizing textual chunks as vector representations (embeddings). These embeddings generally capture the semantics of the chunks, enabling the retriever to perform similarity searches based on content relevance rather than just keyword matching.

According to the sparsity of the embeddings, there are two kinds of encoding methods, i.e., **sparse encoding** and **dense encoding**. The sparse encoding represents text by creating high-dimensional vectors where most elements are zero. The basic sparse encoding is one-hot encoding [50], which represents a word with a high-dimensional vector as large as the vocabulary table size but only marks the value corresponding to the presence of the word as one. The embeddings produced by such encodings are called the one-hot vector. Other common sparse encodings include:

(1) **Bag of Words (BoW) [51].** This encoding improves one-hot encoding by replacing the zero-one counting with the frequency counting. However, BoW ignores the syntax and word order in the documents and focuses on statistical information, thus only expressing limited semantics.

(2) **Term Frequency-Inverse Document Frequency (TF-IDF) [120].** This encoding not only counts the occurrence (frequency) of each word but also adjusts these counts based on how common the word is across all documents (inverse

---

Algorithm 3.1 Building the retriever.

Input: A natural language corpus $D = \{d_1, \ldots, d_n\}$ for building the knowledge database, an encoder $\mathcal{E}$ for encoding chunks.
Output: The index $\mathcal{I}$ and the key-value store $\mathcal{S}$.
  1: $\mathcal{K} = \{\}, \mathcal{V} = \{\};$
  2: **for** $d_i \in D$ **do**
  3:     $c_i^1, \ldots, c_i^m = Chunk(d_i);$ /* Split each data d_i */
  4:     **for** j from 1 to m **do**
  5:        $e_i^j = \mathcal{E}(c_i^j);$ /* Encode each chunk c_i^j */
  6:        Add e_i^j into $\mathcal{K}$ and $c_i^j + c_i^{j+1}$ into $\mathcal{V};$ /* Take next chunk as an exampless */
  7:        The $\mathcal{K}$ and $\mathcal{V}$ persist in the storage (e.g., SSD) if necessary;
  8:     **end for**
  9: **end for**
 10: Build the index $\mathcal{I}$ with $\mathcal{K}$;
 11: Store $\mathcal{K}$ and $\mathcal{V}$ into the key-value store $\mathcal{S}$;
 12: **return** $\mathcal{I}$ and $\mathcal{S}$;

Algorithm 3.2 Query the retriever.

Input: A query input q, an encoder $\mathcal{E}$ for encoding chunks, the index $\mathcal{I}$, the key-value store $\mathcal{S}$, the parameter k.
Output: Top-k nearest neighbor knowledge.
  1: $e = \mathcal{E}(q);$
  2: $\{idx_1, \ldots, idx_k\} = \mathcal{I}.Search(e, k);$ /* Search the top-k nearest neighbors */
  3: $\{v_1, \ldots, v_k\} = \mathcal{S}.Fetch(\{idx_1, \ldots, idx_k\});$ /* Fetch the values of the neighbors */
  4: $\{v_{j_1}, \ldots, v_{j_k}\} = PostProcess(\{v_1, \ldots, v_k\})$
  5: **return** $\{v_{j_1}, \ldots, v_{j_k}\};$

---

document frequency). TF-IDF helps emphasize words that are more descriptive of the document's content. Sparse encoding is an efficient way to encode textual chunks. However, such encodings may not capture deeper semantic meanings.

The dense encoding generates vectors where each dimension can capture a range of semantic features, and most elements are non-zero floating points. The dense embeddings are generally produced by (deep) neural network models,

(1) **BERT [29] and Variants.** Bidirectional Encoder Representation from Transformers (BERT) is a typical pre-trained transformer model, generating dense semantic embeddings that capture the contextual information. Other BERT variants, such as RoBERTa [99], DistilBERT [128], and ELECTRA [19], further improve the semantic representations with advanced learning techniques.

(2) **Siamese Encoders.** This is a type of neural network designed to learn the similarity between inputs, which is usually trained with contrastive learning. Existing state-of-the-art siamese encoders are DPR [79], SimCSE [40].

(3) **LLM-based Encoders.** This type of encoder benefits from the powerful representation capability of LLMs. LLMs, which contain billions of parameters and are pre-trained on vast amounts of data covering a wide range of topics, have advanced semantic language understanding capabilities. Typical LLM-based encoders are text-embedding-ada-002 [113], bge-embedding [156], mxbai-embedding [130].

Compared to sparse encoding, dense encoding leverages deep neural networks, especially transformers [140], to capture broader linguistic and semantic information. Currently, such encodings are widely used in most semantic representation scenarios.

3.1.3 Building the Index. Indexing in the vector database aims to accelerate the search process for data similar to high-dimensional query embedding. Unlike common indexing in databases, indexing in the vector database mainly focuses on supporting efficient approximate nearest neighbor (ANN) search [32, 47, 77] rather than transaction operations like insertion, deletion, and update. The key challenge of indexing is making a good trade-off between search

quality and search efficiency. To solve the challenge, there are various specific optimizations in both algorithmic aspects and systematic aspects to be explored, including choices of similarity metrics, dimension reduction (DR) on embeddings, advanced ANN indexing, system-level optimizations, hardware-aware optimization, and so on. Due to the page limits, this section discusses the optimizations that significantly affect the search quality and efficiency.

Choice of Similarity Metrics. The similarity metrics are the basic components in the retriever, which measures the degree of relevance between query embeddings and chunk embeddings. The similarity metrics would affect the search quality. Typical similarity metrics include cosine similarity, Euclidean similarity, Manhattan distance, and Jaccard similarity.

Dimension Reduction on Embeddings. Reducing the dimensionality of embeddings can improve search efficiency but at the risk of harming the semantic representations. The basic but effective dimension reduction (DR) is the principal component analysis (PCA). The PCA is a simple statistical technique that transforms the original data into a new coordinate system while retaining the most important features. Another popular and advanced dimension reduction is locality-sensitive hashing (LSH). LSH significantly reduces the dimensionality by mapping the data into buckets but preserves the similarity of the original input data. The intuition behind LSH is that the nearest neighbors will be mapped into the same buckets. Unlike LSH, product quantization (PQ) [68] is another popular and effective DR technique for ANN search. The core idea of the PQ is to divide the high-dimensional space into smaller, independently quantized subspaces. Each subspace creates a codebook of different quantized integers to form the representative and compact vectors. The above techniques enable efficient storage and fast approximate search but may lose semantic information. Recent work [17] proposed a new technique named AutoCompressor that reduces the dimension of embeddings by compressing the original context into semantically shorter embeddings.

Advanced ANN Indexing. ANN Indexing generally refers to the methods or structures used to organize and manage data so that the approximate-nearest-neighbor search process is optimized for retrieval quality and retrieval efficiency. This paper will introduce several advanced ANN indexing techniques.

(1) **The InVerted File system with Product Quantization (IVFPQ)** [32] is a simple but effective indexing framework that combines two powerful techniques to enable an efficient and scalable ANN search process. The main idea of IVFPQ is first to cluster the data for coarse-grained partition
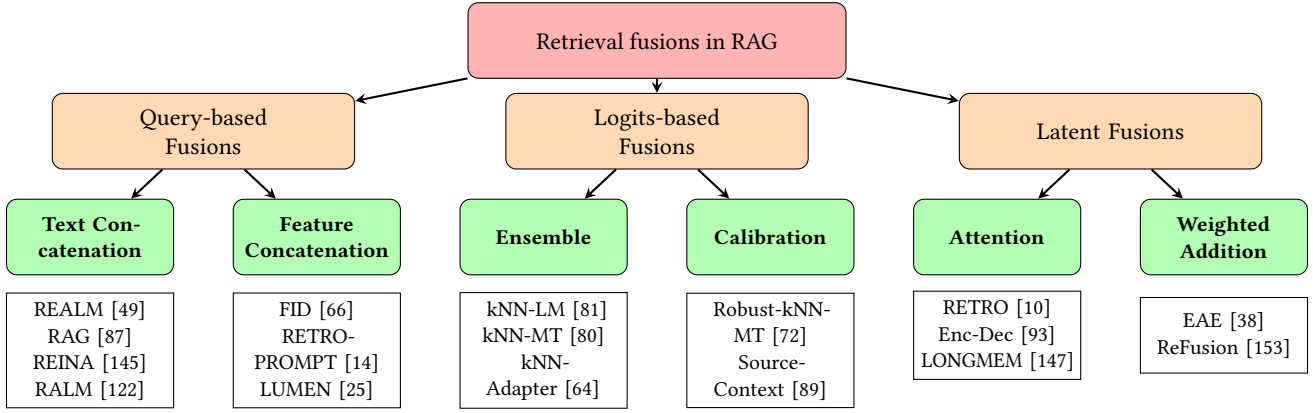
Figure 3: The categories of fusion methods in RAG.

and then to compress the data within each cluster into sub-vectors for fine-grained quantization. The coarse-grained clustering (the IVF component) significantly reduces the search space, while the fine-grained quantization (the PQ component) ensures a high retrieval performance.

(2) **The Hierarchical Navigable Small World (HNSW)** [102] uses a hierarchical graph structure to perform ANN search in high-dimensional spaces efficiently. Specifically, HNSW treats high-dimensional vectors as nodes and connects them with their nearest neighbors. The multi-layer graph structure is determined probabilistically to ensure fewer nodes at higher layers for efficient search.

(3) **Tree-based Indexing** aims to organize high-dimensional vectors in tree-liked structures, such as KD-Trees [123], Ball Trees [62] and VP-Trees [98]. Typical tree-based indexing is Approximate Nearest Neighbors Oh Yeah (Annoy) [135], which uses a forest of trees built based on random projections to separate the vector space into multiple hyperplanes for efficient ANN search.

3.1.4 Building the Datastore with Key-Value Pairs. The datastore used in the vector database is a specialized database that stores and manages data as a collection of key-value pairs, where keys are the unique identifier of high-dimensional embeddings and values are the domain-specific knowledge. Since the amount of the data stored in the datastore may be quite large, the storage engine, such as LMDB [100] or RocksDB [35], should be capable of efficient retrieval and data persistence. The key point in the datastore for ANN search is what should be used to store as values. For example, for question-answer tasks, when adding retrievals to prompts, the naive but effective way is to store the question embedding as the key and question-answer pairs as the value. This can help the generation process as retrievals are used as demonstrations for models. Recent works have proposed various state-of-the-art vector databases, including the indexing and the datastore, such as Milvus [46, 143], FAISS [32, 77], LlamaIndex [95], etc.

3.1.5 Code Demonstrations. Algorithm 3.1 shows detailed steps to build the retriever. Lines 2-8 present the chunking and the encoding

process for a natural language corpus containing multiple documents. In line 6, algorithm 3.1 takes the concatenation of the current chunk and the next chunk as the value. Notably, the choice of value can vary for different tasks. Another practical issue is that the memory cost of all keys and values may exceed the memory capacity of the server in the practical scenario. Thus, it is recommended that the keys and values persist in the storage if necessary.

3.2 Querying the Retriever

This section will explain how to query the pre-built retriever, which basically includes three steps as shown in Figure 2(b): encoding queries, ANN search, and post-processing.

3.2.1 Encoding Queries and ANN Search. To align with the pre-built embedding space, the retriever uses the same encoder to encode queries during the querying stage. The ANN search refers to leveraging the pre-built indexing and datastore to find similar data via approximate nearest neighbor searching algorithms and then retrieve the corresponding values.

Searching the index refers to searching the pre-built index, finding the top-k nearest neighbors, and returning the unique identifiers of k nearest neighbors. The nearest neighbor search process depends on indexing algorithms or structures. Taking IVFPQ as an example, the search process first compares the query embedding with cluster embeddings and selects several candidate clusters for further search. Then, within each cluster, the search process performs the same product quantization on the query embedding and finds the top-k nearest neighbors based on the distance. Finally, the search process merges all nearest neighbor candidates and re-orders all candidates for the final top-k nearest neighbors.

Retrieving values from datastore refers to fetching the corresponding values based on the key identifiers of nearest neighbors.

3.2.2 Post-Processing. The post-processing involves a set of techniques after the initial retrieval step. These techniques aim to refine, enhance, or adapt the retrievals based on the specific task objectives. This section will list some typical post-processing techniques.

Reranking aims to reorder the retrieved knowledge based on task-specific objectives. The intuition is that the knowledge is retrieved based on task-agnostic metrics, such as Euclidean distance.

Algorithm 4.1 Query-based Fusions.

Input: A query input q, top-k nearest neighbor knowledge $\{v_1, \ldots, v_k\}$, an encoder $\mathcal{E}_f$ and a decoder $\mathcal{D}_f$ for feature concatenation, the generator $\mathcal{G}$ for text concatenation.

Output: Generated response y.

1: **if** Use the text concatenation **then**
2:    $x = v_1 \oplus \ldots \oplus v_k \oplus q$; /* Concatenate neighbor texts and query */
3:    $y = \mathcal{G}(x)$;
4: **else**
5:    $e_q = \mathcal{E}_f(q), e_{v_j} = \mathcal{E}_f(v_j), j \in \{1, \ldots, k\}$;
6:    $e_x = e_q \oplus e_{v_1} \oplus \ldots \oplus e_{v_k}$; /* Concatenate embeddings of neighbors and query */
7:    $y = \mathcal{D}_f(e_x)$
8: **end if**
9: **return** y;

Algorithm 4.2 Logits-based Fusions.

Input: A query input q, top-k nearest neighbor knowledge $\{v_1, \ldots, v_k\}$, the generator $\mathcal{G}$.

Output: Generated response y.

1: $y_q = \mathcal{G}(q)$;
2: **for** j from 1 to k **do**
3:    $y_{v_j} = \mathcal{G}(v_j)$
4: **end for**
5: **if** Use ensemble **then**
6:    $y = \lambda \sum_j y_{v_j} + (1 - \lambda) y_q$;
7: **else**
8:    $\lambda_t = Calibrate(y_q, y_{v_1}, \ldots, y_{v_k})$
9:    $y = \lambda_t \sum_j y_{v_j} + (1 - \lambda_t) y_q$;
10: **end if**
11: **return** y;

Existing reranking methods [18, 56, 85, 141] mostly design different architectures or strategies to reorder the retrieved knowledge.

3.2.3 Code Demonstrations. After building the retriever, this section demonstrates the detailed steps of querying the retriever to obtain the top-k nearest neighbor knowledge in algorithm 3.2, including encoding the query (line 1), performing the approximate nearest neighbor search (line 2), and fetching the knowledge for fusion (line 3). These three steps depend on the specific APIs of encoders, indexing, and datastore. After obtaining the top-k retrievals, optimizations for post-processing are applied (line 4).

4 RETRIEVAL FUSIONS

Retrieval fusions refer to how to leverage the retrieved knowledge to improve generators' performance. Basically, there are three types of retrieval fusions: query-based fusions, logits-based fusions, and latent fusions. Figure 3 shows the detailed categorization of fusions and representative works of each retrieval fusion in RAG.

4.1 Query-based Fusion

The simplest and most direct fusion technique is query-based fusion, which integrates the retrieved information with input queries to generate responses. The query-based fusion can be further categorized into two sub-classes according to the type of concatenated information, i.e., text concatenation and feature concatenation.

Text concatenation involves performing query-based fusion with raw texts, making it particularly suitable for contemporary LLMs like GPT-4. These models function as black-box systems with limited interaction capabilities, typically offering only API access to users. Existing works [49, 87, 122] directly concatenate the input with the top-k retrieved sentences/documents to form the query for generators. To better use the in-context learning capability of LLMs, some works [34, 90, 141, 145] design effective prompt templates to integrate retrieved information and inputs. To address the issue of lengthy inputs after concatenating retrievals, recent studies [5, 96, 101, 150, 159] have introduced methods for assigning importance weights to elements within the retrieved knowledge base and filtering out less relevant contexts based on these weights.

The feature concatenation involves merging the encoded retrievals with the input features. A simple yet effective approach is FID [66], which first encodes the retrieved passages into sparse or dense representations and then takes the concatenated features as the input for a generator. The state-of-the-art performance of the FID demonstrates the efficacy of feature concatenation. The follow-up works [25, 48, 67, 97, 127] further improve the FID by jointly tuning the retriever and the encoder, which can enhance the retrieved knowledge's representations. Besides, Chen et al. [14] concatenate the representations of related knowledge as demonstrations for prompt learning, yielding better generalization.

Algorithm 4.1 presents how to leverage query-based fusions to fuse retrieved knowledge. For those using text concatenation [49, 122], algorithm 4.1 first concatenates the retrieved texts and inputs (line 2), then feeds the concatenated input into the generator. Notably, since there is a limit to the maximum input length of existing language models, concatenating too many retrievals would result in a truncation of the concatenated input, which may cut the given input. Therefore, designing the prompt template is the key step for this branch of work. For those using feature concatenation [48, 66], algorithm 4.1 first leverages an encoder to obtain the feature (line 5), then concatenates the feature of input and retrievals (line 6), finally passes the concatenated feature into a decoder model (line 7). This branch of work generally incurs high memory costs due to the long sequence length.

4.2 Logits-based Fusion

The logits-based fusion refers to incorporating the retrieved knowledge into the output layers. Basically, retrieved knowledge would be fed into the same model to obtain the logits for enhancing or calibrating the predictions. Therefore, logits-based fusion can be categorized into two branches, i.e., ensemble-based fusion and calibration-based fusion.

Ensemble-based fusion treats the logits from the retrieved knowledge as part of an ensemble of predictions. Such ensemble-based fusion can significantly improve the generalization and robustness of the model [80, 81, 158]. One notable work of ensemble-based fusion is kNN-LM [81], which aggregates the logits of the top-k nearest neighbors' targets and then interpolates the final predictions. Similar to kNN-LM, Khandelwal et al. [80] propose kNN-MT to enhance the machine translation using retrievals' logits, which is also followed by a branch of works [64, 172].

Algorithm 4.3 Latent Fusions.

Input: A query input q, top-k nearest neighbors $\{v_1, \ldots, v_k\}$, the encoder $\mathcal{E}$, the generator $\mathcal{G}$ containing l pairs of modules $\{(\mathcal{M}_1^A, \mathcal{M}_1^F), \ldots\}$, where $\mathcal{M}_i^A$ and $\mathcal{M}_i^F$ are the attention module and the FFN module at layer i, $\mathcal{M}_i^C$ is the cross-attention module used in attention-based latent fusions.

Output: Generated response y.

1: **if** Use the attention **then**
2:      $h_0^F = q$;
3:      **for** i from 1 to l **do**
4:          $h_i^A = \mathcal{M}_i^A(h_{i-1}^F)$;
5:          $e_{v_1}, \ldots, e_{v_k} = \mathcal{E}(v_1, \ldots, v_k, h_i^A)$
6:          $h_i^R = \mathcal{M}_i^C(h_i^A, e_{v_1}, \ldots, e_{v_k})$; /* Use a cross-attention module to incorporate external knowledge */
7:          $h_i^F = \mathcal{M}_i^F(h_i^R)$
8:      **end for**
9:      $y = LM\_HEAD(h_l^F)$
10: **else**
11:      $e_{v_1}, \ldots, e_{v_k} = \mathcal{E}(v_1, \ldots, v_k)$
12:      $h_0^F = q$;
13:      **for** i from 1 to l **do**
14:          $h_i^A = \mathcal{M}_i^A(h_{i-1}^F)$;
15:          $h_i^R = h_i^A + \frac{1}{k}\sum_j w_j e_{v_j}$ /* Use a weighted sum mechanism to fuse the retrieved knowledge */
16:          $h_i^F = \mathcal{M}_i^F(h_i^R)$
17:      **end for**
18:      $y = LM\_HEAD(h_l^F)$
19: **end if**
20: **return** y;

Different from ensemble-based fusion, calibration-based fusion uses the logits from the retrieved knowledge as a form of calibration for the model's predictions. Specifically, Jiang et al. [72] propose a confidence-enhanced kNN-MT that refines the kNN distribution and interpolation weights with the neural machine translation confidence. Li et al. [89] propose to leverage the source context to calibrate the retrieval-augmented neural machine translation.

Algorithm 4.2 demonstrates the detailed steps of using the logits-based fusion to integrate the retrieved knowledge. This branch of work first treats retrievals as similar data to augment the model (lines 2-4). For ensemble, algorithm 4.2 leverages a hyperparameter to fuse the retrieval logits and the output logits (line 6). For calibration, algorithm 4.2 dynamically determines the parameter based on the retrieval logits and the output logits (line 8). Then, algorithm 4.2 performs the same fusion with the computed parameter (line 9).

4.3 Latent Fusion

The latent fusion investigates merging the retrieved knowledge into the hidden states of generators for a better generation. Based on the introduction method, latent fusion can be further classified into two categories: attention-based and weighted-addition.

One notable contribution of attention-based fusion is the Retrieval-Enhanced Transformer (RETRO) [10]. RETRO represents a pioneering effort in pre-training retrieval-based LLMs, introducing a new cross-attention module to integrate retrieved knowledge directly into the model's hidden states. A significant finding from this work is demonstrating a scaling law for the retrieval database, where

RETRO, with a 2 trillion token database, attains performance comparable to that of major models like GPT-3 and Jurassic-1, albeit with 25 times fewer parameters. Customizing the transformer model in RETRO highlights the potential of pre-trained, retrieval-enhanced architectures in improving the efficiency and scalability of LLMs.

In addition to RETRO, other studies [12, 26, 93, 151, 154] have contributed to the field by leveraging new attention modules to introduce external knowledge. Typically, Li et al. [93] have extended the RETRO model by decoupling the context encoding from the model inference. Wu et al. [154], Wang et al. [147] store the hidden attention keys and values into external memory and retrieve the knowledge from the memory using an attention mechanism.

Due to the high complexity of the attention mechanism, another branch of work adopts lightweight (weighted) additions to introduce retrieved knowledge. Fevry et al. [38] propose the EAE model that retrieves top-k related entities' embeddings from a learnable external memory and adds entities' embeddings to the hidden states of the model. Wu et al. [153] propose ReFusion, which explores various learnable reranking schemes to first re-weight the retrieved knowledge's embeddings, then use weighted addition to incorporate them into the hidden states of the model. Those approaches signify a growing trend towards models that dynamically select and integrate relevant knowledge, paving the way for more sophisticated and nuanced language generation and understanding.

Algorithm 4.3 shows the steps of using latent fusion to introduce the retrieved knowledge into the hidden states of the generator. For attention-based latent fusion, algorithm 4.3 first encodes the retrievals with the output states of the attention module (line 5), then uses a cross-attention module to fuse the retrieval features into the hidden state (line 6). Different from attention-based latent fusion, weighted-addition-based latent fusion adopts a more lightweight way to incorporate retrieved knowledge (lines 10-19). Algorithm 4.3 first encodes the retrievals before feeding them into the generator (line 11), which can be done offline and directly stored as values in the datastore. Then, algorithm 4.3 learns a set of weights to add the retrieval features on the hidden states of generators (line 15).

5 GENERATORS

This section introduces representative generators and retrieval-augmented generators, which are generally pre-trained on large datasets. Existing generators are mostly large language models that adopt or modify the transformer-based architecture [140]. For example, Llama-series models [138, 139], GPT-series models [11, 114, 118, 119], and Gemini-series models [4, 108, 124] remove all encoder modules, retaining only the decoder module, which includes an attention module and a feed-forward network module. Other advanced techniques, such as root mean square layer normalization [165], rotary position embedding [136], and group query attention mechanisms [2], have been incorporated into the design of existing large language models to enhance their performance.

Retrieval-augmented generators typically incorporate new modules into the architecture of existing large language models. They are also pre-trained on a large dataset and an external knowledge database constructed from a vast natural language corpus. These
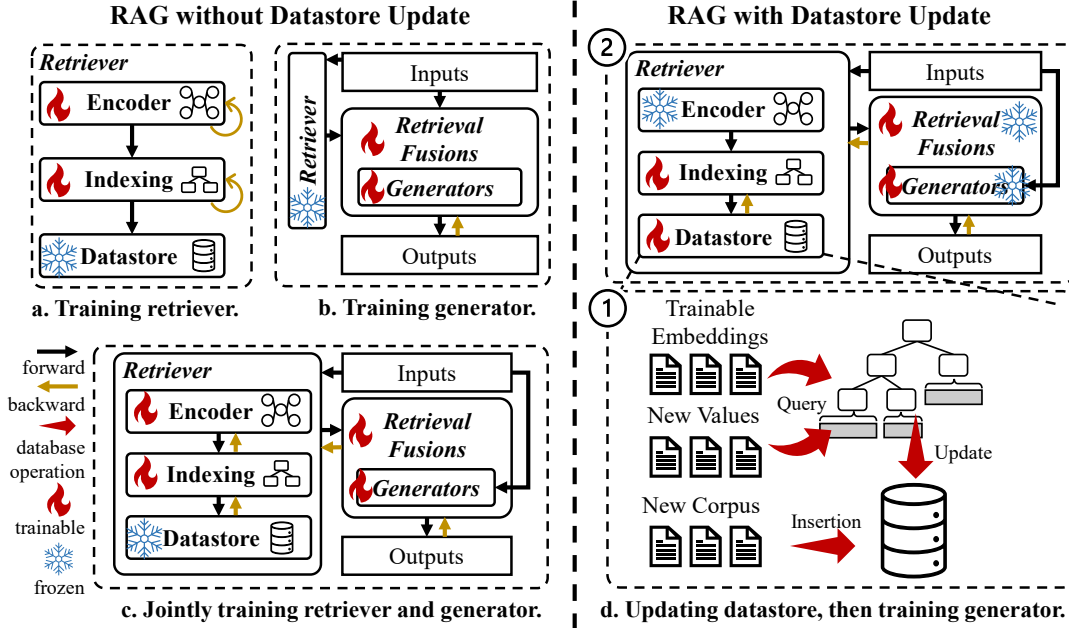
Figure 4: Different RAG training strategies with/without datastore update.

generators mostly leverage latent fusions to incorporate the knowledge into the hidden states of large language models [10, 93, 154], which has been discussed in Section 4.3.

6 RAG TRAINING

This section introduces RAG training, which can be categorized into two main classes: **RAG without datastore update** and **RAG with datastore update**. The former refers to the case where only trainable parameters in each module of RAG would be updated, and the knowledge in the datastore would remain the same during the training stage. The latter refers to the case where the knowledge in the datastore would be updated, then each module's parameters in RAG would be updated in a similar way as the former case.

6.1 RAG without Datastore Update

The goal of training RAG without datastore update is to update the knowledge stored in the short-term memory of generators based on the existing knowledge datastore. As shown in Figure 4 (a)-(c), there are three training cases, i.e., training the retriever, training the generator, and jointly training the retriever and generator.

6.1.1 Training retriever. Considering the case of no datastore update, training the retriever generally refers to training the retriever encoder and rebuilding the indexing. Since sparse encodings typically rely on statistical methods without parameters, training the encoder pertains only to dense encoding methods. Different training methods may have different goals, such as improving the semantic representations, accelerating the encoding process, or learning the domain-specific representations. The first two goals are often achieved by replacing the original encoder with a more powerful or tiny encoder, such as DistilBERT [128] or TinyBERT [75]. The last

requires training the original encoder on the domain-specific corpus using contrastive learning. After training the retriever encoder, the embeddings that serve as keys in the vector database will also change. Thus, all indexes should be rebuilt with new embeddings. Besides, if the encoder remains unchanged, the indexing can be updated using new ANN searching algorithms or re-tuning the hyperparameters. After the retriever is trained, it can be directly incorporated into the RAG without updating the generator.

6.1.2 Training generator. Training the generator involves updating its parameters or those in the retrieval fusion modules. Since the generator is generally an LLM, training the LLM is a resource- and time-consuming process. Fortunately, several parameter-efficient fine-tuning techniques, such as LoRA [57], are proposed to address the fine-tuning problem of LLMs. Although the parameters in the retrieval fusion modules are less than those in the generator, only fine-tuning those parameters may encounter some training problems, such as low convergence and overfitting. Jointly tuning the parameters in the generator and the retrieval fusion modules is a better way to train the generator and the retrieval fusion modules if there are sufficient and powerful resources.

6.1.3 Jointly training the retriever and generator. Apart from independently training the retriever and the generator, jointly training the retriever and the generator can be another good choice for better performance on downstream tasks. The key to this case is to ensure the differentiability from the input to the output during the forward process. Typically, complex indexes, such as FAISS [32], are not a suitable choice during the fine-tuning stage. Existing works generally leverage the complex indexes to pre-select a small subset of nearest neighbors as candidates, then choose the final top-k nearest neighbors by performing the matrix-multiplication operations.

Joint training is an end-to-end optimization that can lead to better coordination between the retriever and the generator and improve the contextual understanding of the generator.

6.2 RAG with Datastore Update

As shown in Figure 4 (d), this scenario involves two stages: updating the knowledge database, then training the retriever and the generator. There are three cases for updating the knowledge database, i.e., updating with trainable embeddings, updating with new values, and updating with new corpus. In the first case, values generally are trainable embeddings and are simultaneously/asynchronously updated with parameters in the RAG [14]. The last two cases usually refer to updating the knowledge database with up-to-date information. Taking question-answer corpus as an example, updating with new values refers to updating the answer to existing questions, while updating with new corpus refers to adding new question-answer pairs. To update the value of existing keys requires first querying the existing key-value pairs and then performing in-place updates. For a new corpus, the datastore first needs to perform insertion operations, then rebuilds or updates the indexes for new keys. After updating the datastore, training the retriever and the generator is similar to RAG without datastore update. However, this training step is not always a necessary step, benefiting to the in-context learning capability of LLMs.

7 TASKS

This section lists several classical tasks in the NLP domain and introduces advanced RAG techniques used to solve these tasks.

7.1 Language Modeling

Language modeling is the task that requires the prediction of the probability distribution of the next word or character given a sequence of words or characters, which is also named the next-token prediction task. Language modeling has become the fundamental task for pre-training large language models, which can measure the models' generation capability using the perplexity metric. The formal definition is as follows: given such a sequence of tokens $x_1, \ldots, x_n$ called *Prefix*, the language modeling task aims to model its probability via next-token prediction,

$$p(x_1, \ldots, x_n) = p(x_1) \cdot \prod_{i=2}^{n} p(x_i|x_1, \ldots, x_{i-1}), \qquad (1)$$

where the conditional probabilities $p(x_i|x_1, \ldots, x_{i-1})$ are modeled by a parameterized language model.

Recent works mainly leverage RAG further to improve language modeling capability in the pre-training stage. A branch of works [10, 93, 147, 154] modifies the architecture of generators by adding a new cross-attention module in each transformer block for introducing retrieval knowledge. The intuition of those works is that given the similar *Prefix*es and their next tokens (retrieving stage), the pre-trained model can calibrate the model's prediction using the cross-attention module to capture the pattern between the next token and prefix (model forwarding stage). Zhong et al. [173] propose to augment the language model with three types of retrieval memories/databases (local memory, long-term memory, and external memory) and optimize the next-token probability distribution with nearest neighbors retrieved from the memories/databases. Another branch of works [49, 64, 81, 122, 160] focuses on augmenting the inputs or outputs of generators with retrievals. Guu et al. [49] and Ram et al. [122] concatenate the retrieved knowledge with inputs and feed the retrieval-augmented inputs into the generators. Other works [64, 81, 160] fuse the logits of inputs as well as retrievals at the final output layer and generate the final probability distribution based on the interpolated results. Those works believe that the concatenated/fused retrievals can provide useful context information on inputs/outputs to improve models' robustness during the pre-training stage. Besides, Doostmohammadi et al. [31] focus on pre-training models with a semantic retriever (BM25) and achieve a better language modeling performance.

7.2 Machine Translation

Machine translation (MT) leverages computational linguistics algorithms to translate text or speech from one language to another automatically. The goal of MT is to produce an accurate and fluent translation, preserving the meaning of the original text while adhering to the grammatical and stylistic norms of the target language. MT systems have evolved from rule-based machine translation (RBMT) to statistical machine translation (SMT) and, more recently, to neural machine translation (NMT). In particular, NMT methods have significantly improved translation quality by leveraging deep learning techniques, which thus will be the focus of this section.

RAG techniques can further enhance MT by incorporating external knowledge into the translation process. The simplest way is to concatenate the similar translation examples into the inputs or fuse the logits of similar translation examples at the output layer. For example, some works [16, 145] retrieve similar translations according to the source text and concatenate corresponding target texts or pairs of source and target texts as examples into inputs. Other works [56, 80, 172] feed the retrieved source text into the models and obtain the logits of the next target tokens, then aggregate all logits to generate the final predictions. Moreover, Jiang et al. [72] and Li et al. [89] use the logits of retrieved examples to calibrate the aggregated logits, improving the robustness of the generation. Another branch of works [173, 174] injects external knowledge into the objective function during the training stage, refining the representation space with similar translations. Besides, Cai et al. [12] encode similar translations and store them as the translation memory, then introduce the knowledge from memory with a cross-attention module. Instead of improving the performance, a branch of work focuses on accelerating the generation efficiency on MT tasks, such as searching from a pre-built subset [27, 107] or a dynamic datastore [22], searching by chunks [104].

7.3 Text Summarization

Text summarization is the process of condensing a larger text document into a shorter version, preserving key information and the overall message. This task can be broadly categorized into two types: extractive summarization, which involves selecting and compiling parts of the original text, and abstractive summarization, which entails rewriting the essence of the text in a new, concise form. The

goal is to produce a coherent and fluent summary that encapsulates the most critical information from the source material.

RAG techniques can significantly enhance text summarization tasks by leveraging external knowledge and similar documents to inform the summarization process. [16, 36, 90, 145] simply concatenates the retrieved similar summaries into inputs to generate summarizations. Instead of concatenating texts, other works fuse features at the intermediate layers by cross-attention [9], or at the output layers by logits ensemble [56]. Besides, Jiang et al. [74] argue that retrieving for every generation may not always be the best choice and propose to retrieve external knowledge during the generation process adaptively.

7.4 Question Answering

Question Answering (QA) is a fundamental task in NLP that involves building systems capable of automatically answering human questions in natural language. QA systems can be broadly classified into two categories: open-domain, where the system answers questions about virtually anything, and closed-domain, focusing on a specific area of knowledge. The primary challenge in QA is understanding the question's intent and retrieving accurate, relevant information from a vast collection of data to provide a concise answer. Due to the page limits, this paper only discusses the works of open-domain QA systems.

RAG techniques combine information retrieval with model-based generation, which is highly suitable for QA systems. In particular, open-domain QA systems usually first require searching for knowledge from the Internet or large-scale databases, then generate the corresponding answers according to the retrieved knowledge. Naturally, given similar questions and corresponding answers as demonstrations which are concatenated into inputs [60, 90, 145], generators in RAG can learn the pattern between questions and answers and infer what answers should be. For some specific QA tasks where a set of reference documents is given, retrievers in RAG would retrieve the relevant documents for concatenation, and then generators in RAG would read the context then generate the final answers via the self-attention mechanism [6, 49, 86, 122], which is similar to solving a reading comprehension problem. Besides, Fabbri et al. [34] focus on designing effective templates for re-organizing the concatenated contexts. Baek et al. [7] leverage the knowledge graph to retrieve the related facts for the input questions, then feed their concatenation and inputs into the generators. Instead of directly concatenating texts, another branch of works focuses on joining the retrieval embeddings with input embeddings for the encoder-decoder models [25, 66, 67, 127].

Some works incorporate the external knowledge in the hidden states or the final logits of generators. For the fusion in the hidden states, the key is what kind of knowledge should be injected, such as entities [26, 38], chunks [10, 142], documents [15]. For the fusion in the logits, most works combine the logits of retrievals and inputs by ensemble techniques [49, 87, 110, 131].

Instead of designing different knowledge fusions for QA systems, existing works also improve QA systems with RAG from other aspects. Some works [45, 94, 116] use retrieved question-answering pairs as extra training data. Some works optimize the retriever module, e.g., improving the keys' representation when building the retriever database [121], replacing the indexing with a pre-trained ranking model [163], or enabling retrieving phrases with two queries [109]. Other works focus on accelerating the generation efficiency of RAG. Jong et al. [24] propose the layer-sparse cross-attention to speed up the decoding. Some works [6, 74, 149] observe that the retrievals may not always provide useful information during the generation process and learn to determine when to retrieve. Moreover, Sun et al. [137] combine the RAG with agents to iteratively reason the final results.

7.5 Information Extraction

Information Extraction (IE) is a critical task in NLP to automatically extract structured information from unstructured and semi-structured text sources. This task encompasses several sub-tasks, including Named Entity Recognition (NER), Entity Linking (EL), Coreference Resolution (CR), Relation Extraction (RE), etc. The goal is to identify and classify key elements from text and understand the relationships between them, thereby converting textual data into a structured format amenable to analysis and interpretation.

With RAG techniques, addressing IE tasks can be significantly improved in terms of not only performance but also interpretability. In NER tasks, Wang et al. [148] first retrieve similar sentences and then concatenate the ranked retrievals for better semantic representations. Ren et al. [126] show that naive RAG may not address Event Argument Extraction (EAE) tasks. Thus, they adopt a sampling-based method to guarantee the same distribution of event labels between retrievals and inputs then concatenate retrieval texts into inputs for better performance in EAE tasks. Table augmentation is also a challenging task, which requires extracting information from tables. Glass et al. [42] propose to extract information in a retrieval-augmented manner.

7.6 Text Classification

Text classification tasks are common in NLP applications. Sentiment analysis, a prominent text classification task in NLP, entails identifying and categorizing the emotional tone conveyed in a text. For example, given a sentence of "I love to watch movies", the analysis models should determine whether it has a positive attitude or a negative attitude. The attitude in sentiment analysis can range from positive to negative or can be neutral, nuanced, and even mixed. The sentiment analysis task is crucial for understanding consumer feedback, monitoring brand reputation, and gaining insights into public opinion on various issues.

RAG techniques can significantly enhance sentiment analysis with different external knowledge fusion strategies. Li et al. [90] concatenate the retrieved options and corresponding prompt-based labels with input options. Other works [14, 48] concatenate the retrieval embeddings with input embeddings before feeding them into the decoder. Some works fuse the retrieval features into the hidden states of generators via cross-attention [15, 147] or ranking-based addition [153]. Besides, other works focus on fusing the logits of retrievals with the output logit using ensemble techniques [161, 167]. Except for knowledge fusions, Min et al. [109] enable locating knowledge in phrases more accurately via two queries.

7.7 Dialogue Systems

Dialogue systems, also known as conversational agents or chatbots, are designed to simulate conversation with human users, either in text or speech form. These systems can be categorized into two main types: task-oriented systems [61], which assist users in completing specific tasks such as booking tickets or ordering food, and open-domain systems, which aim to carry on a general conversation on a wide range of topics [132]. The core challenge in developing effective dialogue systems lies in understanding user intent, maintaining context, and generating coherent, relevant responses.

Existing works improve the dialogue system with RAG mostly via the concatenation-based methods. Some works [16, 83, 92] concatenate the retrieved history conversations with current inputs. Other works [16, 37, 97] first leverage an encoder to encode the history responses, then feed the concatenated embeddings into a decoder to generate new responses.

8 APPLICATIONS

8.1 LLM-based Autonomous Agents

LLM-based autonomous agents are intelligent software systems which leverages the power of LLMs to perform tasks without the need for continuous human intervention [91, 144, 155]. These agents use LLMs as a brain or controller [63], and extend their abilities through multimodal perception [157], tool utilization [129] and external memory [115]. Especially, external long-term memory for agents functions as the knowledge datastore in RAG, which provides agents with the capability to incorporate external knowledge over extended periods. Therefore, applying RAG would be benefit to access a broader range of information, improving agents' decision-making and problem-solving abilities [169]. This section explores how LLM-based agents can leverage RAG from two perspectives.

Using RAG to Retrieve from External Memory. LLM-based agents can utilize RAG to access and retrieve information from their own external memory [52, 105, 170]. This external memory serves as a knowledge base that the agent can draw upon to enhance its understanding and decision-making. When faced with a query or a task, the agent can use RAG to retrieve relevant information from this memory, which is then integrated into the generation process of the LLM. This allows the agent to produce responses or solutions that are informed by a wider range of knowledge, leading to more accurate and contextually relevant outcomes.

The ability to tap into a vast external memory enables the agent to continuously learn and adapt based on new information, making it more effective over time. **Using Tools to Search the Web and RAG for Up-to-Date Information.** In addition to retrieving information from its own memory, an LLM-based agent can use tools to search the web for the most current information [129]. This capability is particularly useful for tasks that require up-to-date knowledge, such as news summarization, market analysis, or responding to rapidly evolving situations. Once the agent retrieves the latest information from the web, it can use RAG to integrate this data into its generation process. By combining the LLM's natural language understanding with real-time data from the web, the agent can generate responses that are not only contextually relevant but also reflect the latest developments. This approach enhances the agent's

ability to provide accurate and timely information, improving its effectiveness in dynamic environments.

In both cases, RAG plays a crucial role in augmenting the capabilities of LLM-based agents by enabling them to access and leverage a wider range of information, whether it's from their own external memory or from real-time sources on the web. This leads to more informed decision-making and enhances the overall performance of the agents.

8.2 Frameworks

Frameworks like Langchain [84] and LLaMAindex [95] pose significant impact on enhancing the practical implementation of RAG. Langchain and LLaMAindex exemplify the integration of sophisticated retrieval mechanisms with generative models, facilitating the seamless incorporation of external data into the language generation process. This section will introduce these two representative RAG frameworks in details.

Langchain is a framework designed to augment the capabilities of language models by integrating them with external knowledge sources and databases. It acts as a middleware that facilitates the interaction between language models and various data retrieval systems, enabling more informed and accurate generation of responses. The core functionality of Langchain involves orchestrating the flow of information from external databases into the generative process of language models, enhancing their ability to leverage context and specific knowledge in their responses. This integration plays a crucial role in enabling language models to perform tasks that require access to up-to-date or detailed information that is not contained within the model's initial training data.

LLaMAindex is a specialized data framework that focuses on organizing and indexing vast amounts of data to improve the retrieval capabilities of language models. This framework supports efficient querying mechanisms, allowing language models to quickly access relevant information from a structured repository. LLaMAindex is designed to be highly scalable and can handle diverse data types, from text documents to structured databases. The indexed data supports a wide range of applications, from simple fact retrieval to complex analytical tasks, making it an indispensable tool for enhancing the information retrieval phase in language models.

Both Langchain and LLaMAindex are deeply connected to the concept of RAG. Langchain enhances RAG by providing a structured way for language models to interact with external databases and knowledge sources during the generation process. On the other hand, LLaMAindex serves as a powerful backend for RAG systems by ensuring that the retrieval process is both fast and relevant. Together, Langchain and LLaMAindex enhance the capabilities of RAG by ensuring that the language models are not only generating text based on their internal knowledge but are also capable of pulling in external data to provide responses that are contextually enriched and informationally robust.

9 DISCUSSION AND FUTURE DIRECTION

Despite the success of the RAG for natural language processing, there are some challenges that should be considered. This paper highlights these challenges to inspire future research and provides possible future research directions in RAG for NLP.

9.1 Retrieval Quality

The retrieval quality refers to improving the relevance of the information retrieved in RAG, which involves the following four key factors to be designed. The first consideration is **determining the optimal key** to use in the vector database. This process typically involves subjective decision-making and requires human effort to design effectively. The naive idea is to choose inputs for the given tasks, treating each task as a QA problem.

The second is **the choice of embedding model**. After determining the key, the next step is leveraging embedding models to convert text into vector representations. Models such as BERT [29], RoBERTa [99], or domain-specific embeddings can be crucial to determine how well nuances and contextual meanings are captured. Adapting the embedding model to better suit specific types of data or queries can significantly enhance retrieval quality. This requires training the model on domain-specific corpora that include the types of queries and documents the system will encounter.

Thirdly, **designing effective similarity metrics** is also crucial to improve retrieval quality. The goal of similarity metrics is to measure the relevance between the query and the retrieved information. Some classical similarity metrics, such as cosine similarity or Euclidean distance, used for ranking in the recommender system can also be used in RAG [44]. Apart from these metrics, some works explored more complex similarity metrics, such as optimal transport distance [21], to obtain a task-specific similarity.

Finally, **approximate nearest neighbor (ANN) searching** is also a key step in determining what knowledge should be returned as nearest neighbors. Advanced ANN searching aims to accelerate the retrieval efficiency at the cost of sacrificing the retrieval quality. Choosing a suitable ANN algorithm, such as product quantization [68] or HNSW [102], requires a good trade-off between retrieval efficiency and retrieval quality. All of these factors collectively contribute to the retrieval quality of the retriever.

9.2 RAG Efficiency

RAG efficiency is crucial for downstream NLP applications, which limits the volume of data that can be retrieved. There are two simple ways to guarantee RAG efficiency without new algorithms, i.e., reducing the volume of data or adding more powerful computing and memory resources. However, the former may impact the retrieval quality, while the latter requires more resource cost.

RAG efficiency encompasses the efficiency of the retriever and the efficiency of retrieval fusions. Retriever efficiency refers to the time cost of retrieving relevant information, which can be divided into three parts, i.e., encoding time, ANN searching time, and data fetching time of the datastore. It is unnecessary to jointly optimize all three components as the bottleneck would vary from different database sizes. For smaller retrieval databases, such as those with fewer than 1 million entries, the encoding phase is often the primary bottleneck, as the vector database can be all stored in the memory. Several topics, such as model quantization [8, 82], distillation [30, 75], or model pruning [39], are used to accelerate the encoding.

In contrast, for larger databases, the time cost of searching in the index and fetching data from the datastore becomes the major bottleneck, as the searching is over a considerable amount of data, and the fetching involves I/O overheads. In this case, efficient ANN searching algorithms [32, 47, 77] and system-level optimizations [73, 76] are the main focus.

Retrieval fusion efficiency, which aims to enhance the inference efficiency when integrating retrievals, is worth to be optimized for improving the RAG efficiency. For example, the computational overhead of query-based fusion is often non-negligible due to the long sequence length. Some works, such as Fid-light [55] and Re-Fusion [153], mainly target reducing the computations while integrating the retrieved information.

9.3 Choices of Fusions

This paper introduces three kinds of retrieval fusions, where each fusion is worth further exploring. Query-based fusions concatenate the texts or embeddings of retrieved knowledge with inputs. These methods have better interpretability and are easy to apply even only when the API of LLMs is provided. However, concatenation leads to a long sequence of inputs, thus resulting in a large computational overhead in the attention and truncation of inputs. Some works [5, 153] aim to improve efficiency when integrating retrievals, while others [9, 147] focus on improving the efficiency when increasing the model input length.

Conversely, latent-based fusions amalgamate information at a deeper, more abstract level, which may capture more nuanced relationships between the retrieved information and the query. However, these fusions significantly lack interpretability and often require pre-training or fine-tuning to adjust the retrieval embeddings or reweight the retrievals. Therefore, enhancing the interpretability of such latent-based fusions is also worth exploring in the future.

Logits-based fusions incorporate information at the decision level, thereby offering a potentially more flexible and robust integration of data from various sources. Nonetheless, these fusions may oversimplify the fusion process, diminishing the richness of the retrieved information by reducing them to logit values. Meanwhile, such fusions require performing all inference of retrievals, which is also a time-consuming process.

Apart from applying one kind of fusion in practical applications, combining different fusions is also worth exploring for better performance. These fusion methods are not mutually exclusive, as they focus on augmenting the different stages of generators, i.e., inputs, hidden states, and outputs. Besides, during the generation, when to fuse retrieved knowledge is also a significant problem worthy of further exploration [103].

9.4 RAG Training

As introduced in Section 6, RAG training includes two branch of works, RAG with/without datastore update. For RAG without datastore update, the main challenge is how to jointly optimize all parameters in RAG. This may involves new loss functions with multiple objectives, new optimizations for efficient tuning parameters in retriever and generator, or other training strategies.

For RAG with datastore update, one challenge is how to align the retrieval representations with the generator's representations. Although the time cost of the update operation in datastore cannot be ignored, some works [14] reduce the update frequency by asynchronously updating, thus achieving the alignment of knowledge representation and model's representation. Another challenge is

when to retrain/fine-tune the generator in RAG when new corpus is added. Due to the in-context learning capability of exisitng LLM-based generators and high training overhead, retraining/fine-tuning the generator or directly inferring the generator becomes a challenging choice for different scenarios. Recently, some efficient training strategies [28, 57] have been proposed to accelerate the fine-tuning process, which can be taken into considerations.

9.5 Cross-Modality Retrieval

Retrieving cross-modality information in NLP tasks can greatly enhance the quality and richness of the representations, leading to improved performance. First, cross-modality information, such as combining text with images, videos, or audio, provides a richer context to the content [58]. For instance, when language is ambiguous, accompanying images can clarify meanings difficult to convey through text alone. Second, different modalities can contribute various types of information that are not accessible from a single source. For example, visual data can provide spatial, color, and action cues, while textual data can offer detailed descriptions, emotions, or abstract concepts. Combining these can lead to a more comprehensive understanding of the data. Moreover, Models trained on multi-modal data typically exhibit increased robustness and generalizability [152]. These models are adept at associating information across diverse inputs, mitigating overfitting to the peculiarities of a single modality. This attribute is particularly valuable in real-world applications of NLP, such as in autonomous vehicles, where systems must interpret textual information from signs or dialogues and sensory data from the surrounding environment to make informed decisions. Furthermore, multi-modal data can resolve ambiguities that cannot be resolved within a single modality. For example, the phrase "bank" can refer to either a financial institution or the side of a river, and visual context can help disambiguate this. Last, human communication is inherently multi-modal, incorporating elements such as gestures, facial expressions, and tone of voice. Systems capable of processing multiple modes of communication can interact with humans in a manner that is both more natural and intuitive. In conclusion, integrating cross-modality information in RAG for NLP tasks not only enhances the richness and quality of data representations but also significantly improves the systems' comprehension, interaction capabilities, and adaptability to diverse applications.

10 CONCLUSION

In this survey, we delve into the development of RAG within the field of natural language processing. First, this paper introduces the components of RAG and their functionalities. Subsequently, this paper elaborates on each step involved in retriever, discussing the diverse techniques. Furthermore, this paper categorizes the retrieval fusions, evaluating the strengths and weaknesses inherent of each retrieval fusion techniques. Besides, this paper discusses the RAG training, including RAG with/without datastore update. Then, this paper explores how RAG can be adapted for various NLP tasks and provides practical applications of RAG in real-world scenarios. Conclusively, this paper identifies ongoing challenges and suggests directions for future research to foster advancements in this evolving area.

REFERENCES

[1] Samira Abnar, Mostafa Dehghani, Behnam Neyshabur, and Hanie Sedghi. 2022. Exploring the Limits of Large Scale Pre-training. In *The Tenth International Conference on Learning Representations (ICLR)*.

[2] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, 4895–4901.

[3] Badr AlKhamissi, Millicent Li, Asli Celikyilmaz, Mona T. Diab, and Marjan Ghazvininejad. 2022. A Review on Language Models as Knowledge Bases. *CoRR* abs/2204.06031 (2022).

[4] Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Slav Petrov, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy P. Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy, Michael Isard, Paul Ronald Barham, Tom Hennigan, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, Ryan Doherty, Eli Collins, Clemens Meyer, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, George Tucker, Enrique Piqueras, Maxim Krikun, Iain Barr, Nikolay Savinov, Ivo Danihelka, Becca Roelofs, Anaïs White, Anders Andreassen, Tamara von Glehn, Lakshman Yagati, Mehran Kazemi, Lucas Gonzalez, Misha Khalman, Jakub Sygnowski, and et al. 2023. Gemini: A Family of Highly Capable Multimodal Models. *CoRR* abs/2312.11805 (2023).

[5] Md. Adnan Arefeen, Biplob Debnath, and Srimat Chakradhar. 2023. LeanContext: Cost-Efficient Domain-Specific Question Answering Using LLMs. *CoRR* abs/2309.00841 (2023).

[6] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. *CoRR* abs/2310.11511 (2023).

[7] Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023. Knowledge-Augmented Language Model Prompting for Zero-Shot Knowledge Graph Question Answering. *CoRR* abs/2306.04136 (2023).

[8] Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jin Jin, Xin Jiang, Qun Liu, Michael R. Lyu, and Irwin King. 2021. BinaryBERT: Pushing the Limit of BERT Quantization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL/IJCNLP)*. Association for Computational Linguistics, 4334–4348.

[9] Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew R. Gormley. 2023. Unlimiformer: Long-Range Transformers with Unlimited Length Input. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*.

[10] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. 2022. Improving Language Models by Retrieving from Trillions of Tokens. In *Proceedings of the 39th International Conference on Machine Learning (ICML) (Proceedings of Machine Learning Research)*, Vol. 162. 2206–2240.

[11] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*.

[12] Deng Cai, Yan Wang, Huayang Li, Wai Lam, and Lemao Liu. 2021. Neural Machine Translation with Monolingual Translation Memory. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL/IJCNLP)*. 7307–7318.

[13] Junying Chen, Qingcai Chen, Dongfang Li, and Yutao Huang. 2022. SeDR: Segment Representation Learning for Long Documents Dense Retrieval. *CoRR* abs/2211.10841 (2022).

[14] Xiang Chen, Lei Li, Ningyu Zhang, Xiaozhuan Liang, Shumin Deng, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2022. Decoupling Knowledge from Memorization: Retrieval-augmented Prompt Learning. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*.

[15] Xin Cheng, Yankai Lin, Xiuying Chen, Dongyan Zhao, and Rui Yan. 2023. Decouple knowledge from paramters for plug-and-play language modeling. In *Findings of the Association for Computational Linguistics (ACL)*. 14288–14308.

[16] Xin Cheng, Di Luo, Xiuying Chen, Lemao Liu, Dongyan Zhao, and Rui Yan. 2023. Lift Yourself Up: Retrieval-augmented Text Generation with Self-Memory. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*.

[17] Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting Language Models to Compress Contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 3829–3846.

[18] Yung-Sung Chuang, Wei Fang, Shang-Wen Li, Wen-tau Yih, and James R. Glass. 2023. Expand, Rerank, and Retrieve: Query Reranking for Open-Domain Question Answering. In *Findings of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, 12131–12147.

[19] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *8th International Conference on Learning Representations (ICLR)*. OpenReview.net.

[20] Pierre Colombo, Telmo Pessoa Pires, Malik Boudiaf, Dominic Culver, Rui Melo, Caio Corro, André F. T. Martins, Fabrizio Esposito, Vera Lúcia Raposo, Sofia Morgado, and Michael Desa. 2024. SaulLM-7B: A pioneering Large Language Model for Law. *CoRR* abs/2403.03883 (2024).

[21] Yufei Cui, Ziquan Liu, Yixin Chen, Yuchen Lu, Xinyue Yu, Xue (Steve) Liu, Tei-Wei Kuo, Miguel Rodrigues, Chun Jason Xue, and Antoni B. Chan. 2023. Retrieval-Augmented Multiple Instance Learning. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*.

[22] Yuhan Dai, Zhirui Zhang, Qiuzhi Liu, Qu Cui, Weihua Li, Yichao Du, and Tong Xu. 2023. Simple and Scalable Nearest Neighbor Machine Translation. In *The Eleventh International Conference on Learning Representations (ICLR)*.

[23] David Dale, Elena Voita, Loïc Barrault, and Marta R. Costa-jussà. 2023. Detecting and Mitigating Hallucinations in Machine Translation: Model Internal Workings Alone Do Well, Sentence Similarity Even Better. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, 36–50.

[24] Michiel de Jong, Yury Zemlyanskiy, Joshua Ainslie, Nicholas FitzGerald, Sumit Sanghai, Fei Sha, and William W. Cohen. 2023. FiDO: Fusion-in-Decoder optimized for stronger performance and faster inference. In *Findings of the Association for Computational Linguistics (ACL)*. 11534–11547.

[25] Michiel de Jong, Yury Zemlyanskiy, Nicholas FitzGerald, Joshua Ainslie, Sumit Sanghai, Fei Sha, and William W. Cohen. 2023. Pre-computed memory or on-the-fly encoding? A hybrid approach to retrieval augmentation makes the most of your compute. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*. 7329–7342.

[26] Michiel de Jong, Yury Zemlyanskiy, Nicholas FitzGerald, Fei Sha, and William W. Cohen. 2022. Mention Memory: incorporating textual knowledge into Transformers through entity mention attention. In *The Tenth International Conference on Learning Representations (ICLR)*.

[27] Hiroyuki Deguchi, Taro Watanabe, Yusuke Matsui, Masao Utiyama, Hideki Tanaka, and Eiichiro Sumita. 2023. Subset Retrieval Nearest Neighbor Machine Translation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*. 174–189.

[28] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*.

[29] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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 (NAACL-HLT)*. 4171–4186.

[30] Zixiang Ding, Guoqing Jiang, Shuai Zhang, Lin Guo, and Wei Lin. 2023. SKD-BERT: Compressing BERT via Stochastic Knowledge Distillation. In *Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press, 7414–7422.

[31] Ehsan Doostmohammadi, Tobias Norlund, Marco Kuhlmann, and Richard Johansson. 2023. Surface-Based Retrieval Reduces Perplexity of Retrieval-Augmented Language Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*. 521–529.

[32] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The Faiss library. *CoRR* abs/2401.08281 (2024).

[33] explosion. 2016. *Spacy*. https://spacy.io/

[34] Alexander R. Fabbri, Patrick Ng, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2020. Template-Based Question Generation from Retrieved Sentences for Improved Unsupervised Question Answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*. 4508–4513.

[35] Facebook. 2013. *RocksDB*. https://github.com/facebook/rocksdb

[36] Angela Fan and Claire Gardent. 2022. Generating Full Length Wikipedia Biographies: The Impact of Gender Bias on the Retrieval-Based Generation of Women Biographies. *CoRR* abs/2204.05879 (2022).

[37] Angela Fan, Claire Gardent, Chloé Braud, and Antoine Bordes. 2021. Augmenting Transformers with KNN-Based Composite Memory for Dialog. *Trans. Assoc. Comput. Linguistics* 9 (2021), 82–99.

[38] Thibault Févry, Livio Baldini Soares, Nicholas FitzGerald, Eunsol Choi, and Tom Kwiatkowski. 2020. Entities as Experts: Sparse Memory Access with Entity Supervision. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 4937–4951.

[39] Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Hassan Sajjad, Preslav Nakov, Deming Chen, and Marianne Winslett. 2021. Compressing Large-Scale Transformer-Based Models: A Case Study on BERT. *Trans. Assoc. Comput. Linguistics* 9 (2021), 1061–1080.

[40] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 6894–6910.

[41] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2023. Retrieval-Augmented Generation for Large Language Models: A Survey. *CoRR* abs/2312.10997 (2023).

[42] Michael R. Glass, Xueqing Wu, Ankita Rajaram Naik, Gaetano Rossiello, and Alfio Gliozzo. 2023. Retrieval-Based Transformer for Table Augmentation. In *Findings of the Association for Computational Linguistics (ACL)*. 5635–5648.

[43] Hongyu Gong, Yelong Shen, Dian Yu, Jianshu Chen, and Dong Yu. 2020. Recurrent Chunking Mechanisms for Long-Text Machine Reading Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, 6751–6761.

[44] Asela Gunawardana and Guy Shani. 2009. A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. *J. Mach. Learn. Res.* 10 (2009), 2935–2962.

[45] Daya Guo, Duyu Tang, Nan Duan, Ming Zhou, and Jian Yin. 2019. Coupling Retrieval and Meta-Learning for Context-Dependent Semantic Parsing. In *Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL)*. 855–866.

[46] Rentong Guo, Xiaofan Luan, Long Xiang, Xiao Yan, Xiaomeng Yi, Jigao Luo, Qianya Cheng, Weizhi Xu, Jiarui Luo, Frank Liu, Zhenshan Cao, Yanliang Qiao, Ting Wang, Bo Tang, and Charles Xie. 2022. Manu: A Cloud Native Vector Database Management System. *Proc. VLDB Endow.* 15, 12 (2022), 3548–3561.

[47] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating Large-Scale Inference with Anisotropic Vector Quantization. In *Proceedings of the 37th International Conference on Machine Learning (ICML) (Proceedings of Machine Learning Research)*, Vol. 119. PMLR, 3887–3896.

[48] Zhicheng Guo, Sijie Cheng, Yile Wang, Peng Li, and Yang Liu. 2023. Prompt-Guided Retrieval Augmentation for Non-Knowledge-Intensive Tasks. In *Findings of the Association for Computational Linguistics (ACL)*. 10896–10912.

[49] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Retrieval Augmented Language Model Pre-Training. In *Proceedings of the 37th International Conference on Machine Learning (ICML) (Proceedings of Machine Learning Research)*, Vol. 119. 3929–3938.

[50] David Harris and Sarah Harris. 2010. *Digital design and computer architecture*. Morgan Kaufmann.

[51] Zellig S Harris. 1954. Distributional structure. *Word* 10, 2-3 (1954), 146–162.

[52] Kostas Hatalis, Despina Christou, Joshua Myers, Steven Jones, Keith Lambert, Adam Amos-Binks, Zohreh Dannenhauer, and Dustin Dannenhauer. 2023. Memory Matters: The Need to Improve Long-Term Memory in LLM-Agents. In *Proceedings of the AAAI Symposium Series*, Vol. 2. 277–280.

[53] Qiyuan He, Yizhong Wang, and Wenya Wang. 2024. Can Language Models Act as Knowledge Bases at Scale? *CoRR* abs/2402.14273 (2024).

[54] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training Compute-Optimal Large Language Models. *CoRR* abs/2203.15556 (2022).

[55] Sebastian Hofstätter, Jiecao Chen, Karthik Raman, and Hamed Zamani. 2023. FiD-Light: Efficient and Effective Retrieval-Augmented Text Generation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. ACM, 1437–1447.

[56] Nabil Hossain, Marjan Ghazvininejad, and Luke Zettlemoyer. 2020. Simple and Effective Retrieve-Edit-Rerank Text Generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*. 2532–2538.

[57] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *The Tenth International Conference on Learning Representations (ICLR)*.

[58] Xuming Hu. 2023. Multimodal Named Entity Recognition and Relation Extraction with Retrieval-Augmented Strategy. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. ACM, 3488.

[59] Yucheng Hu and Yuxing Lu. 2024. RAG and RAU: A Survey on Retrieval-Augmented Language Model in Natural Language Processing. *CoRR* abs/2404.19543 (2024).

[60] Jie Huang, Wei Ping, Peng Xu, Mohammad Shoeybi, Kevin Chen-Chuan Chang, and Bryan Catanzaro. 2023. RAVEN: In-Context Learning with Retrieval Augmented Encoder-Decoder Language Models. *CoRR* abs/2308.07922 (2023).

[61] Qiushi Huang, Shuai Fu, Xubo Liu, Wenwu Wang, Tom Ko, Yu Zhang, and Lilian H. Y. Tang. 2023. Learning Retrieval Augmentation for Personalized Dialogue Generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2523–2540.

[62] Qiang Huang and Anthony K. H. Tung. 2023. Lightweight-Yet-Efficient: Revitalizing Ball-Tree for Point-to-Hyperplane Nearest Neighbor Search. In *39th IEEE International Conference on Data Engineering (ICDE)*. IEEE, 436–449.

[63] Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. 2024. Understanding the planning of LLM agents: A survey. *CoRR* abs/2402.02716 (2024).

[64] Yangsibo Huang, Daogao Liu, Zexuan Zhong, Weijia Shi, and Yin Tat Lee. 2023. kNN-Adapter: Efficient Domain Adaptation for Black-Box Language Models. *CoRR* abs/2302.10879 (2023).

[65] Shonosuke Ishiwatari, Jingtao Yao, Shujie Liu, Mu Li, Ming Zhou, Naoki Yoshinaga, Masaru Kitsuregawa, and Weijia Jia. 2017. Chunk-based Decoder for Neural Machine Translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, 1901–1912.

[66] Gautier Izacard and Edouard Grave. 2021. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. 874–880.

[67] Gautier Izacard, Patrick S. H. Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot Learning with Retrieval Augmented Language Models. *J. Mach. Learn. Res.* 24 (2023), 251:1–251:43.

[68] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product Quantization for Nearest Neighbor Search. *IEEE Trans. Pattern Anal. Mach. Intell.* 33, 1 (2011), 117–128.

[69] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of Hallucination in Natural Language Generation. *ACM Comput. Surv.* 55, 12 (2023), 248:1–248:38.

[70] Ziwei Ji, Zihan Liu, Nayeon Lee, Tiezheng Yu, Bryan Wilie, Min Zeng, and Pascale Fung. 2023. RHO: Reducing Hallucination in Open-domain Dialogues with Knowledge Grounding. In *Findings of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, 4504–4522.

[71] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. *CoRR* abs/2310.06825 (2023).

[72] Hui Jiang, Ziyao Lu, Fandong Meng, Chulun Zhou, Jie Zhou, Degen Huang, and Jinsong Su. 2022. Towards Robust k-Nearest-Neighbor Machine Translation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 5468–5477.

[73] Wenqi Jiang, Shuai Zhang, Boran Han, Jie Wang, Bernie Wang, and Tim Kraska. 2024. PipeRAG: Fast Retrieval-Augmented Generation via Algorithm-System Co-design. *CoRR* abs/2403.05676 (2024).

[74] Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active Retrieval Augmented Generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 7969–7992.

[75] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for Natural Language Understanding. In *Findings of the Association for Computational Linguistics (EMNLP) (Findings of ACL)*, Vol. EMNLP 2020. Association for Computational Linguistics, 4163–4174.

[76] Chao Jin, Zili Zhang, Xuanlin Jiang, Fangyue Liu, Xin Liu, Xuanzhe Liu, and Xin Jin. 2024. RAGCache: Efficient Knowledge Caching for Retrieval-Augmented Generation. *CoRR* abs/2404.12457 (2024).

[77] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-Scale Similarity Search with GPUs. *IEEE Trans. Big Data* 7, 3 (2021), 535–547.

[78] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models. *CoRR* abs/2001.08361 (2020).

[79] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 6769–6781.

[80] Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Nearest Neighbor Machine Translation. *CoRR* abs/2010.00710 (2020).

[81] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through Memorization: Nearest Neighbor Language Models. In *The 8th International Conference on Learning Representations (ICLR)*.

[82] Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. 2021. I-BERT: Integer-only BERT Quantization. In *Proceedings of the 38th International Conference on Machine Learning (ICML) (Proceedings of Machine Learning Research)*, Vol. 139. PMLR, 5506–5518.

[83] Brendan King and Jeffrey Flanigan. 2023. Diverse Retrieval-Augmented In-Context Learning for Dialogue State Tracking. In *Findings of the Association for Computational Linguistics (ACL)*. 5570–5585.

[84] LangChain. 2023. *LangChain*. https://www.langchain.com/

[85] Angeliki Lazaridou, Elena Gribovskaya, Wojciech Stokowiec, and Nikolai Grigorev. 2022. Internet-augmented language models through few-shot prompting for open-domain question answering. *CoRR* abs/2203.05115 (2022).

[86] Kyungjae Lee, Sang-eun Han, Seung-won Hwang, and Moontae Lee. 2023. When to Read Documents or QA History: On Unified and Selective Open-domain QA. In *Findings of the Association for Computational Linguistics (ACL)*. 6420–6432.

[87] Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*.

[88] Huayang Li, Yixuan Su, Deng Cai, Yan Wang, and Lemao Liu. 2022. A Survey on Retrieval-Augmented Text Generation. *CoRR* abs/2202.01110 (2022).

[89] Xuanhong Li, Peng Li, and Po Hu. 2023. Revisiting Source Context in Nearest Neighbor Machine Translation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[90] Xiaonan Li, Kai Lv, Hang Yan, Tianyang Lin, Wei Zhu, Yuan Ni, Guotong Xie, Xiaoling Wang, and Xipeng Qiu. 2023. Unified Demonstration Retriever for In-Context Learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*. 4644–4668.

[91] Yuanchun Li, Hao Wen, Weijun Wang, Xiangyu Li, Yizhen Yuan, Guohong Liu, Jiacheng Liu, Wenxing Xu, Xiang Wang, Yi Sun, Rui Kong, Yile Wang, Hanfei Geng, Jian Luan, Xuefeng Jin, Zilong Ye, Guanjing Xiong, Fan Zhang, Xiang Li, Mengwei Xu, Zhijun Li, Peng Li, Yang Liu, Ya-Qin Zhang, and Yunxin Liu. 2024. Personal LLM Agents: Insights and Survey about the Capability, Efficiency and Security. *CoRR* abs/2401.05459 (2024).

[92] Yunhao Li, Yunyi Yang, Xiaojun Quan, and Jianxing Yu. 2021. Retrieve & Memorize: Dialog Policy Learning with Multi-Action Memory. In *Findings of the Association for Computational Linguistics (ACL/IJCNLP)*. 447–459.

[93] Zonglin Li, Ruiqi Guo, and Sanjiv Kumar. 2022. Decoupled Context Processing for Context Augmented Language Modeling. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*.

[94] Bill Yuchen Lin, Kangmin Tan, Chris Miller, Beiwen Tian, and Xiang Ren. 2022. Unsupervised Cross-Task Generalization via Retrieval Augmentation. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*.

[95] Jerry Liu. 2022. *LlamaIndex*. https://doi.org/10.5281/zenodo.1234

[96] Junyi Liu, Liangzhi Li, Tong Xiang, Bowen Wang, and Yiming Qian. 2023. TCRA-LLM: Token Compression Retrieval Augmented Large Language Model for Inference Cost Reduction. In *Findings of the Association for Computational Linguistics (EMNLP)*. 9796–9810.

[97] Shuai Liu, Hyundong Cho, Marjorie Freedman, Xuezhe Ma, and Jonathan May. 2023. RECAP: Retrieval-Enhanced Context-Aware Prefix Encoder for Personalized Dialogue Response Generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*. 8404–8419.

[98] Shi-guang Liu and Yin-wei Wei. 2015. Fast nearest neighbor searching based on improved VP-tree. *Pattern Recognit. Lett.* 60-61 (2015), 8–15.

[99] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR* abs/1907.11692 (2019).

[100] LMDB. 2014. *LMDB*. https://github.com/LMDB/lmdb

[101] Xiaozhong Lyu, Stefan Grafberger, Samantha Biegel, Shaopeng Wei, Meng Cao, Sebastian Schelter, and Ce Zhang. 2023. Improving Retrieval-Augmented Large Language Models via Data Importance Learning. *CoRR* abs/2307.03027 (2023).

[102] Yury A. Malkov and Dmitry A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 42, 4 (2020), 824–836.

[103] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*. 9802–9822.

[104] Pedro Henrique Martins, Zita Marinho, and André F. T. Martins. 2022. Chunk-based Nearest Neighbor Machine Translation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 4228–4245.

[105] Kai Mei, Zelong Li, Shuyuan Xu, Ruosong Ye, Yingqiang Ge, and Yongfeng Zhang. 2024. AIOS: LLM Agent Operating System. *CoRR* abs/2403.16971 (2024).

[106] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and Editing Factual Associations in GPT. In *Advances in Neural Information*

Processing Systems 35 (NeurIPS).

[107] Yuxian Meng, Xiaoya Li, Xiayu Zheng, Fei Wu, Xiaofei Sun, Tianwei Zhang, and Jiwei Li. 2022. Fast Nearest Neighbor Machine Translation. In *Finding of the Association for Computational Linguistics (ACL)*. 555–565.

[108] Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Cristian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, and et al. 2024. Gemma: Open Models Based on Gemini Research and Technology. *CoRR* abs/2403.08295 (2024).

[109] Sewon Min, Weijia Shi, Mike Lewis, Xilun Chen, Wen-tau Yih, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2023. Nonparametric Masked Language Modeling. In *Findings of the Association for Computational Linguistics (ACL)*. 2097–2118.

[110] Aaron Mueller, Kanika Narang, Lambert Mathias, Qifan Wang, and Hamed Firooz. 2023. Meta-training with Demonstration Retrieval for Efficient Few-shot Learning. In *Findings of the Association for Computational Linguistics (ACL)*. 6049–6064.

[111] Ewa Muszynska. 2016. Graph- and surface-level sentence chunking. In *Proceedings of the ACL 2016 Student Research Workshop*. Association for Computational Linguistics, 93–99.

[112] NLTK. 2001. *NLTK*. https://www.nltk.org/

[113] OpenAI. 2022. *Text-Emb-Ada*. https://platform.openai.com/docs/guides/embeddings

[114] OpenAI. 2023. GPT-4 Technical Report. *CoRR* abs/2303.08774 (2023).

[115] Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. 2023. MemGPT: Towards LLMs as Operating Systems. *CoRR* abs/2310.08560 (2023).

[116] Bhargavi Paranjape, Matthew Lamm, and Ian Tenney. 2022. Retrieval-guided Counterfactual Generation for QA. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*. 1670–1686.

[117] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. 2019. Language Models as Knowledge Bases?. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, 2463–2473.

[118] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *OpenAI blog* (2018).

[119] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[120] Anand Rajaraman and Jeffrey David Ullman. 2011. *Data Mining*. Cambridge University Press, 1–17.

[121] Ori Ram, Liat Bezalel, Adi Zicher, Yonatan Belinkov, Jonathan Berant, and Amir Globerson. 2023. What Are You Token About? Dense Retrieval as Distributions Over the Vocabulary. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*. 2481–2498.

[122] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-Context Retrieval-Augmented Language Models. *Trans. Assoc. Comput. Linguistics* 11 (2023), 1316–1331.

[123] Parikshit Ram and Kaushik Sinha. 2019. Revisiting kd-tree for Nearest Neighbor Search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis (Eds.). ACM, 1378–1388.

[124] Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy P. Lillicrap, Jean-Baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, Ioannis Antonoglou, Rohan Anil, Sebastian Borgeaud, Andrew M. Dai, Katie Millican, Ethan Dyer, Mia Glaese, Thibault Sottiaux, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, James Molloy, Jilin Chen, Michael Isard, Paul Barham, Tom Hennigan, Ross McIlroy, Melvin Johnson, Johan Schalkwyk, Eli Collins, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, Clemens Meyer, Gregory Thornton, Zhen Yang, Henryk Michalewski, Zaheer Abbas, Nathan Schucher, Ankesh Anand, Richard Ives, James Keeling, Karel Lenc, Salem Haykal, Siamak Shakeri, Pranav Shyam, Aakanksha Chowdhery, Roman Ring, Stephen Spencer, Eren Sezener, and et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *CoRR* abs/2403.05530 (2024).

[125] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint*

Conference on Natural Language Processing (EMNLP-IJCNLP). 3980–3990.

[126] Yubing Ren, Yanan Cao, Ping Guo, Fang Fang, Wei Ma, and Zheng Lin. 2023. Retrieve-and-Sample: Document-level Event Argument Extraction via Hybrid Retrieval Augmentation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*. 293–306.

[127] Devendra Singh Sachan, Siva Reddy, William L. Hamilton, Chris Dyer, and Dani Yogatama. 2021. End-to-End Training of Multi-Document Reader and Retriever for Open-Domain Question Answering. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*. 25968–25981.

[128] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR* abs/1910.01108 (2019).

[129] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*.

[130] Darius Koenig Julius Lipp Sean Lee, Aamir Shakir. 2024. *Open Source Strikes Bread - New Fluffy Embeddings Model*. https://www.mixedbread.ai/blog/mxbai-embed-large-v1

[131] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023. REPLUG: Retrieval-Augmented Black-Box Language Models. *CoRR* abs/2301.12652 (2023).

[132] Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval Augmentation Reduces Hallucination in Conversation. In *Findings of the Association for Computational Linguistics (EMNLP)*. Association for Computational Linguistics, 3784–3803.

[133] Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Kumar Tanwani, Heather Cole-Lewis, Stephen Pfohl, Perry Payne, Martin Seneviratne, Paul Gamble, Chris Kelly, Nathaneal Schärli, Aakanksha Chowdhery, Philip Andrew Mansfield, Blaise Agüera y Arcas, Dale R. Webster, Gregory S. Corrado, Yossi Matias, Katherine Chou, Juraj Gottweis, Nenad Tomasev, Yun Liu, Alvin Rajkomar, Joelle K. Barral, Christopher Semturs, Alan Karthikesalingam, and Vivek Natarajan. 2023. Large Language Models Encode Clinical Knowledge. *Nature* 620, 7972 (2023), 172–180.

[134] Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Le Hou, Kevin Clark, Stephen Pfohl, Heather Cole-Lewis, Darlene Neal, Mike Schaekermann, Amy Wang, Mohamed Amin, Sami Lachgar, Philip Andrew Mansfield, Sushant Prakash, Bradley Green, Ewa Dominowska, Blaise Agüera y Arcas, Nenad Tomasev, Yun Liu, Renee Wong, Christopher Semturs, S. Sara Mahdavi, Joelle K. Barral, Dale R. Webster, Gregory S. Corrado, Yossi Matias, Shekoofeh Azizi, Alan Karthikesalingam, and Vivek Natarajan. 2023. Towards Expert-Level Medical Question Answering with Large Language Models. *CoRR* abs/2305.09617 (2023).

[135] Spotify. 2017. *Annoy*. https://github.com/spotify/annoy

[136] Jianlin Su, Murtadha H. M. Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. RoFormer: Enhanced transformer with Rotary Position Embedding. *Neurocomputing* 568 (2024), 127063.

[137] Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Heung-Yeung Shum, and Jian Guo. 2023. Think-on-Graph: Deep and Responsible Reasoning of Large Language Model with Knowledge Graph. *CoRR* abs/2307.07697 (2023).

[138] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *CoRR* abs/2302.13971 (2023).

[139] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *CoRR* abs/2307.09288 (2023).

[140] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*. 5998–6008.

[141] Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry W. Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc V. Le, and Thang Luong. 2023.

FreshLLMs: Refreshing Large Language Models with Search Engine Augmentation. *CoRR* abs/2310.03214 (2023).

[142] Boxin Wang, Wei Ping, Peng Xu, Lawrence McAfee, Zihan Liu, Mohammad Shoeybi, Yi Dong, Oleksii Kuchaiev, Bo Li, Chaowei Xiao, Anima Anandkumar, and Bryan Catanzaro. 2023. Shall We Pretrain Autoregressive Language Models with Retrieval? A Comprehensive Study. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 7763–7786.

[143] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, Kun Yu, Yuxing Yuan, Yinghao Zou, Jiquan Long, Yudong Cai, Zhenxiang Li, Zhifeng Zhang, Yihua Mo, Jun Gu, Ruiyi Jiang, Yi Wei, and Charles Xie. 2021. Milvus: A Purpose-Built Vector Data Management System. In *SIGMOD '21: International Conference on Management of Data*. ACM, 2614–2627.

[144] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. 2024. A survey on large language model based autonomous agents. *Frontiers Comput. Sci.* 18, 6 (2024), 186345.

[145] Shuohang Wang, Yichong Xu, Yuwei Fang, Yang Liu, Siqi Sun, Ruochen Xu, Chenguang Zhu, and Michael Zeng. 2022. Training Data is More Valuable than You Think: A Simple and Effective Method by Retrieving from Training Data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*. 3170–3179.

[146] Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. 2023. Knowledge Editing for Large Language Models: A Survey. *CoRR* abs/2310.16218 (2023).

[147] Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. 2023. Augmenting Language Models with Long-Term Memory. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*.

[148] Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021. Improving Named Entity Recognition by External Context Retrieving and Cooperative Learning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL/IJCNLP)*. 1800–1812.

[149] Yile Wang, Peng Li, Maosong Sun, and Yang Liu. 2023. Self-Knowledge Guided Retrieval Augmentation for Large Language Models. In *Findings of the Association for Computational Linguistics (EMNLP)*. 10303–10315.

[150] Zhiruo Wang, Jun Araki, Zhengbao Jiang, Md. Rizwan Parvez, and Graham Neubig. 2023. Learning to Filter Context for Retrieval-Augmented Generation. *CoRR* abs/2311.08377 (2023).

[151] Zichao Wang, Weili Nie, Zhuoran Qiao, Chaowei Xiao, Richard G. Baraniuk, and Anima Anandkumar. 2023. Retrieval-based Controllable Molecule Generation. In *The Eleventh International Conference on Learning Representations, (ICLR)*.

[152] Mike Wu and Noah D. Goodman. 2018. Multimodal Generative Models for Scalable Weakly-Supervised Learning. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*. 5580–5590.

[153] Shangyu Wu, Ying Xiong, Yufei Cui, Xue Liu, Buzhou Tang, Tei-Wei Kuo, and Chun Jason Xue. 2024. ReFusion: Improving Natural Language Understanding with Computation-Efficient Retrieval Representation Fusion. *CoRR* abs/2401.02993 (2024).

[154] Yuhuai Wu, Markus Norman Rabe, DeLesley Hutchins, and Christian Szegedy. 2022. Memorizing Transformers. In *The Tenth International Conference on Learning Representations (ICLR)*.

[155] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, and Tao Gui. 2023. The Rise and Potential of Large Language Model Based Agents: A Survey. *CoRR* abs/2309.07864 (2023).

[156] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighof. 2023. C-Pack: Packaged Resources To Advance General Chinese Embedding. *CoRR* abs/2309.07597 (2023).

[157] Junlin Xie, Zhihong Chen, Ruifei Zhang, Xiang Wan, and Guanbin Li. 2024. Large Multimodal Agents: A Survey. *CoRR* abs/2402.15116 (2024).

[158] Ying Xiong, Xin Yang, Linjing Liu, Ka-Chun Wong, Qingcai Chen, Yang Xiang, and Buzhou Tang. 2023. EARA: Improving Biomedical Semantic Textual Similarity with Entity-Aligned Attention and Retrieval Augmentation. In *Findings of the Association for Computational Linguistics (EMNLP)*. Association for Computational Linguistics, 8760–8771.

[159] Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2023. RECOMP: Improving Retrieval-Augmented LMs with Compression and Selective Augmentation. *CoRR* abs/2310.04408 (2023).

[160] Frank F. Xu, Uri Alon, and Graham Neubig. 2023. Why do Nearest Neighbor Language Models Work?. In *Proceedings of the 40th International Conference on Machine Learning (ICML) (Proceedings of Machine Learning Research)*, Vol. 202. 38325–38341.

[161] Guoxin Yu, Lemao Liu, Haiyun Jiang, Shuming Shi, and Xiang Ao. 2023. Retrieval-Augmented Few-shot Text Classification. In *Findings of the Association for Computational Linguistics (EMNLP)*. 6721–6735.

[162] Hao Yu, Aoran Gan, Kai Zhang, Shiwei Tong, Qi Liu, and Zhaofeng Liu. 2024. Evaluation of Retrieval-Augmented Generation: A Survey. *CoRR* abs/2405.07437 (2024).

[163] Zichun Yu, Chenyan Xiong, Shi Yu, and Zhiyuan Liu. 2023. Augmentation-Adapted Retriever Improves Generalization of Language Models as Generic Plug-In. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*. 2421–2436.

[164] Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, Weng Lam Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen, Zhiyuan Liu, Peng Zhang, Yuxiao Dong, and Jie Tang. 2023. GLM-130B: An Open Bilingual Pre-trained Model. In *The Eleventh International Conference on Learning Representations (ICLR)*. OpenReview.net.

[165] Biao Zhang and Rico Sennrich. 2019. Root Mean Square Layer Normalization. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*. 12360–12371.

[166] Hongbo Zhang, Junying Chen, Feng Jiang, Fei Yu, Zhihong Chen, Guiming Chen, Jianquan Li, Xiangbo Wu, Zhiyi Zhang, Qingying Xiao, Xiang Wan, Benyou Wang, and Haizhou Li. 2023. HuatuoGPT, Towards Taming Language Model to Be a Doctor. In *Findings of the Association for Computational Linguistics (EMNLP)*. Association for Computational Linguistics, 10859–10885.

[167] Jianyi Zhang, Aashiq Muhamed, Aditya Anantharaman, Guoyin Wang, Changyou Chen, Kai Zhong, Qingjun Cui, Yi Xu, Belinda Zeng, Trishul Chilimbi, and Yiran Chen. 2023. ReAugKD: Retrieval-Augmented Knowledge Distillation For Pre-trained Language Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*. 1128–1136.

[168] Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, Siyuan Cheng, Ziwen Xu, Xin Xu, Jia-Chen Gu, Yong Jiang, Pengjun Xie, Fei Huang, Lei Liang, Zhiqiang Zhang, Xiaowei Zhu, Jun Zhou, and Huajun Chen. 2024. A Comprehensive Study of Knowledge Editing for Large Language Models. *CoRR* abs/2401.01286 (2024).

[169] Yadong Zhang, Shaoguang Mao, Tao Ge, Xun Wang, Adrian de Wynter, Yan Xia, Wenshan Wu, Ting Song, Man Lan, and Furu Wei. 2024. LLM as a Mastermind: A Survey of Strategic Reasoning with Large Language Models. *CoRR* abs/2404.01230 (2024).

[170] Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2024. A Survey on the Memory Mechanism of Large Language Model based Agents. *CoRR* abs/2404.13501 (2024).

[171] Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, and Bin Cui. 2024. Retrieval-Augmented Generation for AI-Generated Content: A Survey. *CoRR* abs/2402.19473 (2024).

[172] Xin Zheng, Zhirui Zhang, Junliang Guo, Shujian Huang, Boxing Chen, Weihua Luo, and Jiajun Chen. 2021. Adaptive Nearest Neighbor Machine Translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL/IJCNLP)*. 368–374.

[173] Zexuan Zhong, Tao Lei, and Danqi Chen. 2022. Training Language Models with Memory Augmentation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 5657–5673.

[174] Wenhao Zhu, Jingjing Xu, Shujian Huang, Lingpeng Kong, and Jiajun Chen. 2023. INK: Injecting kNN Knowledge in Nearest Neighbor Machine Translation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*. 15948–15959.