OneGen: Efficient One-Pass Unified Generation and Retrieval for LLMs

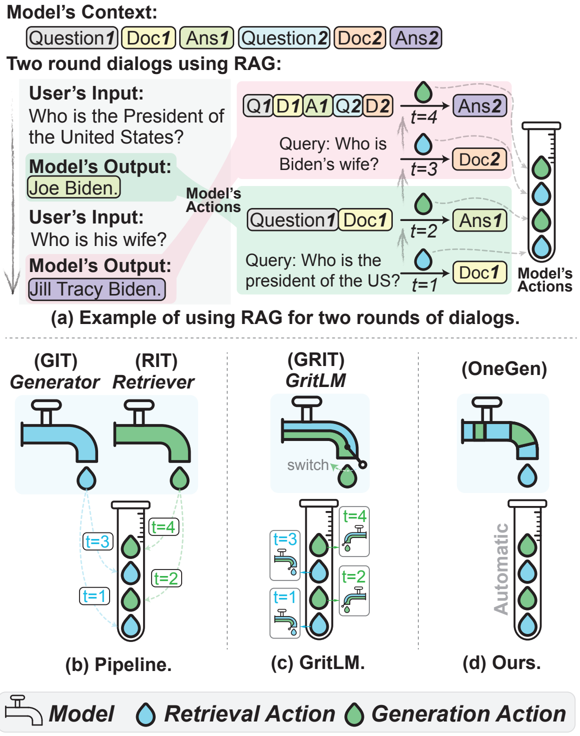
Jintian Zhang♠♡* , **Cheng Peng**♣♡* , **Mengshu Sun**♣♡, **Xiang Chen**♠♡, **Lei Liang**♣♡,
Zhiqiang Zhang♣♡, **Jun Zhou**♣♡†, **Huajun Chen**♠♡, **Ningyu Zhang**♠♡†
♠Zhejiang University ♣Ant Group
♡Zhejiang University - Ant Group Joint Laboratory of Knowledge Graph
{zhangjintian,zhangningyu}@zju.edu.cn, jun.zhoujun@antgroup.com
https://github.com/zjunlp/OneGen

Abstract

Despite the recent advancements in Large Language Models (LLMs), which have significantly enhanced the generative capabilities for various NLP tasks, LLMs still face limitations in directly handling retrieval tasks. However, many practical applications demand the seamless integration of both retrieval and generation. This paper introduces a novel and efficient **One**-pass **Gen**eration and retrieval framework (**OneGen**), designed to improve LLMs' performance on tasks that require both generation and retrieval. The proposed framework bridges the traditionally separate training approaches for generation and retrieval by incorporating retrieval tokens generated autoregressively. This enables a single LLM to handle both tasks simultaneously in a unified forward pass. We conduct experiments on two distinct types of composite tasks, RAG and Entity Linking, to validate the pluggability, effectiveness, and efficiency of OneGen in training and inference. Furthermore, our results show that integrating generation and retrieval within the same context preserves the generative capabilities of LLMs while improving retrieval performance. To the best of our knowledge, OneGen is the first to enable LLMs to conduct vector retrieval during the generation.

1 Introduction

In the era of Large Language Models (LLMs), many Natural Language Processing (NLP) tasks can be reduced to generation, allowing them to be addressed by a single LLM (Zhao et al., 2023; Qin et al., 2023; OpenAI, 2023; Zeng et al., 2024). While LLMs excel in language generation, they still suffer from hallucinations (e.g., factual inaccuracies), stemming from their exclusive reliance on the parametric knowledge they contain (Zhang et al., 2023b; Yao et al., 2023; Tonmoy et al., 2024).



(a) Example of using RAG for two rounds of dialogs.

(b) Pipeline. (c) GritLM. (d) Ours.

Figure 1: Comparison of Three Methods for RAG Task. (a) Two round dialogs using RAG (Retrieve and Generate twice each). (b) Pipeline approach requiring the deployment of two separate models for retrieval and generation, (c) GritLM (Muennighoff et al., 2024) utilizing a single model with a switching mechanism to integrate retrieval and generation, (d) OneGen (Ours) performing both functions automatically in the same model and the same context.

One promising approach is Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Jiang et al., 2023d; Asai et al., 2024; Mao et al., 2024; Gao et al., 2023), which augments the input by retrieving relevant passages based on the query either before or during generation. Other methods (Ding et al., 2024a; Luo et al., 2023a) anchor LLM generation to an external knowledge base through Entity Linking (EL) during or after generation. These systems typically rely on a *retriever* at various stages of generation.

[*] Equal Contribution.
[†] Corresponding Author.

Findings of the Association for Computational Linguistics: EMNLP 2024, pages 4088–4119
November 12-16, 2024 ©2024 Association for Computational Linguistics

However, due to the separate training paradigms for generation and retrieval, most prior work (Muennighoff et al., 2024) employs a separate model for text embedding. However, this pipeline approach has several drawbacks: *i)* Deploying and maintaining two separate models introduces additional hardware overhead and increases maintenance costs. *ii)* The separation of models creates two distinct representational spaces, limiting interaction between the retriever and generator (e.g., LLM) to text (i.e., query). As a result, whether the query is generated by the LLM or input directly by the user, it requires an additional forward pass through the retriever, increasing inference computational costs. *iii)* In multi-turn dialogues, as illustrated in Figure 1(a), query rewriting is required for follow-up questions like "Who is his wife?". This rewriting adds inference overhead and risks error propagation if inaccurate. *iv)* Additionally, the pipeline approach is difficult to optimize end-to-end and requires large amounts of training data, while end-to-end optimization has been shown to yield significant benefits (Lin et al., 2024).

Our work introduces an efficient **One**-pass unified **Gen**eration and retrieval (**OneGen**) framework to enable an arbitrary LLM to generate and retrieve in one single forward pass. Inspired by the latest success in LLM for text embedding (Wang et al., 2024), we expand the original vocabulary by adding special tokens (i.e. *retrieval tokens*) and allocate the retrieval task to retrieval tokens generated in an autoregressive manner. During training, *retrieval tokens* only participate in representation fine-tuning through contrastive learning (van den Oord et al., 2018; Rendle et al., 2009), whereas other output tokens are trained using language model objectives. At inference time, we use *retrieval tokens* for efficient retrieving on demand.

Unlike previous pipeline approaches, which require at least two models for retrieval and generation (as shown in Figure 1(b)), OneGen unifies both tasks into a single model, eliminating the need for a separate retriever. (Muennighoff et al., 2024) presents Generative Representational Instruction Tuning (GRIT), which aligns with this approach by training one LLM to handle both generative and embedding tasks through different prompts and attention mechanisms, as depicted by the *"switch"* in Figure 1(c). However, GRIT still necessitates independent forward passes for generation and retrieval tasks, reducing efficiency for tasks that intertwine generation and retrieval.

We evaluate the effectiveness of our method on two main tasks that require both generation and retrieval: RAG (including single-hop QA which needs single-retrieval and multi-hop QA which needs multi-retrieval) and Entity Linking (EL). Empirical results show OneGen outperforms the previous pipeline solutions as well as GRIT where applicable. Specifically, OneGen achieves +1.5pt improvement on average with four Single-hop QA datasets on top of Self-RAG (Asai et al., 2024), +3.3pt F1 on average with two Multi-hop QA datasets under three different 7B-based LLMs, and +3.2pt accuracy on average with 6 out-of-domain entity linking datasets, with less training data. Moreover, further analysis demonstrates OneGen can enhance retrieval capability when jointly trained, with no sacrifice in generation capability. In addition, we demonstrate superior inference speed and memory consumption of OneGen compared with other LLM alternatives, particularly as retrieval frequency increases. In summary, our work makes the following **contributions**:

i) We propose a training-efficiency, inference-efficiency, and pluggable framework OneGen that is particularly suitable for tasks interleaved with generation and retrieval. *ii)* Our model, fine-tuned on less training data, demonstrates superior performance on six RAG datasets and six entity linking datasets on average. *iii)* We demonstrate the efficiency of OneGen at inference, highlighting a significant speed improvement as the length of query increases or retrieval frequency increases, compared to other LLM alternatives. *iv)* From the perspective of methodology, OneGen is an extension of Generative Instruction Tuning (GIT) and Representative Instruction Tuning (RIT) (as shown in Figure 1(b)). *v)* We contribute to communities by releasing our dataset as well as code.

2 Preliminaries and Related Works

Most text-based tasks can be reduced to generation, retrieval, or combination of the two. We first introduce several hybrid tasks and their common solutions in § 2.1. Then, we introduce the three roles of tokens in LLMs in § 2.2. Finally, we further explain the motivation of our method in § 2.3.

2.1 Generation & Retrieval

For NLP problem related to generation or retrieval, a user input or a query $u = \{u_1, ..., u_n\}$ and optionally document corpus $\mathcal{K} = \{d_i\}_{i=1}^{\|\mathcal{K}\|}$ are given

Task	Input	$t = 1$	$t = 2$
Generation	u	$y = G(u)$	–
Retrieval	$u, \mathcal{K}$	$\varepsilon = R(u, \mathcal{K})$	–
$R \to G$	$u, \mathcal{K}$	$\varepsilon = R(u, \mathcal{K})$	$y = G(u, \varepsilon)$
$G \to R$	$u, \mathcal{K}$	$y = G(u)$	$\varepsilon = R(y, \mathcal{K})$
Unified	$u, \mathcal{K}$	$y, \varepsilon = OneGen(u, \mathcal{K})$	

Table 1: Comparison of different tasks and our unified solution.

(e.g., wiki articles), the end goal of the task is to generate sequence output $y = \{y_1, ..., y_m\}$ or the most relevant documents ϵ from $\mathcal{K}$ with respect to u or both. We also assume that each $d_i \in \varepsilon$ is aligned to a subsequence or a whole sequence of tokens in u. We summarize the steps and typical input, output for generation, retrieval, and two hybrid tasks in Table 1.

R $\to$ G Task leverages retrieval results to drive generation. In the simplest format, a dense retrieval model (e.g., a dense passage retriever, DPR) is used to retrieve a collection of relevant documents ε given user input u at t=1; ε are then used as additional context when generating the target sequence using a generator (e.g. LLM) at t=2. Retrieval Argumented Generation (RAG) is a classic example of $R \to G$ task. Though there are some efforts in training the two model end-to-end predate the LLM era (Lewis et al., 2020), most recent work use an off-the-shelf-retriever such as Contriever (Izacard et al., 2022), BM25, or search engine (Jiang et al., 2023d). Furthermore, this task can involve multiple iterations of retrieval and generation, such as in multi-hop reasoning datasets like 2WIKI (Ho et al., 2020) and HotpotQA (Yang et al., 2018).

G $\to$ R Task outputs retrieved documents relevant to user query in addition to generated content and are widely encountered in Information Retrieval (IR). A prominent example task is Entity Linking (EL), which involves locating mentions and disambiguating these surface forms into entities in some Knowledge Base (KB). Early EL methods (Hoffmann et al., 2011) treat EL as decomposed subtasks, such as Mention Detection (MD) and Entity Disambiguation (ED), and solve them in sequence. More recent works manage to frame EL as an end-to-end task, such as sequence generation (Cao et al., 2021b), question answering (Zhang et al., 2022), retrieve augmented generation (Xiao et al., 2023), and sequence tagging problem (Broscheit, 2019; Ayoola et al., 2022), which outperform the early pipeline approach. For the generative EL paradigm, MD can be modeled as a generation task where entities in the original sentences are generated; ED is a typical retrieval task of retrieving the most relevant entity from the KB given a mention span.

2.2 Roles of tokens in LLMs

A token x_i is the basic unit processed by an LLM. Token in the input of an LLM serves three different roles: *1) generating the next token*, noted as $role(x_i)$ = `GEN`; *2) providing context information*, noted as $role(x_i)$ = `CTX`; and *3) representing a sentence*, noted as $role(x_i)$ = `RET`. Recent works (Wang et al., 2024; Muennighoff et al., 2024) use the hidden state of the last token as the sentence representation.

2.3 Motivation

Recent years have seen a rise in using LLMs to handle complex hybrid tasks, replacing traditional NLP model pipelines. Before LLMs, end-to-end approaches offered advantages for combining generation and retrieval tasks, reducing error propagation compared to pipelines and potentially improving efficiency with single-pass inference. However, earlier solutions are often task-specific and lack generalization across hybrid tasks. For instance, in generative EL, methods like constrained decoding (Cao et al., 2021b) are used to retrieve entities efficiently. Our work addresses the absence of a unified LLM framework for hybrid tasks, stemming from separate training approaches for generation and retrieval tasks, which typically use distinct objectives and datasets.

3 OneGen: One-Pass Generation and Retrieval for LLMs

We introduce a **One**-pass **Gen**eration and retrieval framework (**OneGen**) for fine-tuning LLMs on generation, retrieval, or hybrid tasks, as shown in Figure 2. Our core idea is to integrate generation and retrieval to the same context by allocating the retrieval task to *retrieval tokens* generated in an autoregressive manner, thus enabling LLM to perform both tasks in a single forward pass.

3.1 Overview

Notation. To ensure clarity and precision in our subsequent discussions, we standardize the notation used in Table 1. Define the dataset $\mathcal{D} = \{s_i\}_{i=1}^{|\mathcal{D}|}$, which consists of $|\mathcal{D}|$ sentences s of varying lengths, with each sentence $s = \{x_i\}_{i=0}^{|s|}$ com-

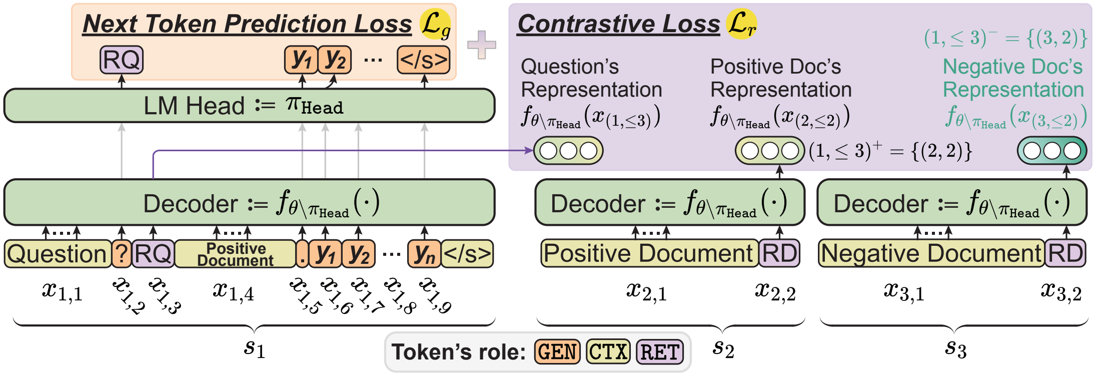

Next Token Prediction Loss $\mathcal{L}_g$
RQ
y_1 y_2 ⋯ </s>
LM Head := π_{Head}
Contrastive Loss $\mathcal{L}_r$
$(1, \leq 3)^- = \{(3,2)\}$
Question's Representation $f_{\theta\backslash\pi_{\text{Head}}}(x_{(1,\leq 3)})$
Positive Doc's Representation $f_{\theta\backslash\pi_{\text{Head}}}(x_{(2,\leq 2)})$
Negative Doc's Representation $f_{\theta\backslash\pi_{\text{Head}}}(x_{(3,\leq 2)})$
$(1, \leq 3)^+ = \{(2,2)\}$
Decoder := $f_{\theta\backslash\pi_{\text{Head}}}(\cdot)$
Decoder := $f_{\theta\backslash\pi_{\text{Head}}}(\cdot)$
Decoder := $f_{\theta\backslash\pi_{\text{Head}}}(\cdot)$
Question ? RQ Positive Document . y_1 y_2 ⋯ y_n </s>
Positive Document RD
Negative Document RD
$x_{1,1}$ $x_{1,2}$ $x_{1,3}$ $x_{1,4}$ $x_{1,5}$ $x_{1,6}$ $x_{1,7}$ $x_{1,8}$ $x_{1,9}$
$x_{2,1}$ $x_{2,2}$
$x_{3,1}$ $x_{3,2}$
s_1
Token's role: GEN CTX RET
s_2
s_3

Figure 2: The training framework of unified **One**-pass **Gen**eration and retrieval (**OneGen**), illustrated using RAG. Detailed training process for other tasks can be found in Figure 6 of Appendix.

prising $|s|$ tokens x. Let $x_{i,j}$ denote the j-th token of the i-th sentence in the dataset $\mathcal{D}$, and define $x_{i,\leq j}$ as $\{x_{i,1}, x_{i,2}, \ldots, x_{i,j}\}$. We can distinguish the symbols u, y, and d defined in Table 1 based on the role of tokens x within the sentence s. Specifically, y corresponds to the segment of the s where $role(x) = \texttt{GEN}$, u corresponds to the segment where $role(x) = \texttt{CTX}$, and if all tokens x in a sentence s have $role(x) = \texttt{CTX}$, then it corresponds to d. Given the instruction dataset $\mathcal{I}$, where $s = \{u, y\} \in \mathcal{I}$, we have $\mathcal{D} = \mathcal{I} \cup \mathcal{K}$.

Design. Retrieval requires encoding both the query and the document within the same representational space. *Our core idea is to incorporate query encoding into the generation process.* Thus we use the same LLM for encoding both the query and the document, without altering the model structure, such as the attention mechanism, unlike the approach taken by GritLM. Specifically, for query encoding, we introduce a special token $x_i = \texttt{[RQ]}$, where $role(x_i) = \texttt{RET}$. This token is generated by the LLM and used as input to represent the query. However, assigning $role(x_i) = \texttt{RET}$ prevents the generation of the next token x_{i+1} if $role(x_{i+1}) = \texttt{GEN}$. To address this, we also introduce a `<CON>` token during data reconstruction, ensuring the continuation of the generation process.

Inference. At inference time, the documents to be retrieved are encoded offline by the trained LLM using the template "`{document}[RD]`", where $role(\texttt{[RD]}) = \texttt{RET}$. Then the trained LLM autoregressively generates tokens based on the user's input until it encounters the `[RQ]` token. The logits corresponding to the `[RQ]` token are then used for retrieval. Depending on the task requirements, the retrieved content may be concatenated with the context, potentially along with the `<CON>` token, before continuing with the inference until the generation is complete.

3.2 Train

Data Reconstruction. We augment the standard generation output with retrieval tokens wherever retrieval is needed. This makes our framework easily pluggable to existing methods. Generally, we insert `[RQ]` to sentence s for query representation. In particular, if the query span is explicit, we add optional tokens `<LOC>` and `</LOC>` to assist in locating the position of the query. The augmented sequence is $s = \{x_{\leq i}, \texttt{<LOC>}, x_{i+1}, \ldots, x_j, \texttt{</LOC>}, \texttt{[RQ]}, \texttt{<CON>}, x_{(>j)}\}$. The token `<CON>` enables LLMs to generate continuously and it must be included if and only if $role(x_{j+1}) = \texttt{GEN}$, i.e., generation is required after retrieval. For each document $x \in \mathcal{K}$, `[RD]` is usually appended to the end of the document to represent the document. In particular, we can add `[RD]` for each end of the sentence in a document $x \in \mathcal{K}$ to get the fine-grained representation. Figure 8 and Figure 11 in appendix show two different examples for reconstructed document .

Training Objective. The optimization is only performed for tokens $x_i \in s$ where $role(x_i) \in \{\texttt{GEN}, \texttt{RET}\}$. A simple application of OneGen in the RAG task is illustrated in Figure 2. Note that, $role(x_i) = \texttt{RET}$ iff $x_i \in \{\texttt{[RQ]}, \texttt{[RD]}\}$ (highlight in purple in Figure 2). For tokens where $role(x_i) = \texttt{GEN}$ (highlight in orange), optimization employs $\mathcal{L}_g$:

$$\mathcal{L}_g = \sum_{i=1}^{|\mathcal{D}|}\sum_{j=1}^{|s_i|} \ell_g(f_{\theta\backslash\pi_{\text{Head}}}(x_{(i,\leq j)}), \pi_{\text{Head}}) \cdot \mathbb{1}_g(x_{i,j}).$$

Here, θ is the LLM parameter. $\pi_{\text{Head}} \in \mathbb{R}^{N \times d}$ denotes the expanded vocabulary (i.e., LM Head), consists of N d-dimensional vectors.

$f_{\theta\backslash\pi_{\mathsf{Head}}}(x_{(i,\leq j)}) \in \mathbb{R}^d$ denotes a d-dimensional vector generated by the LLM without LM Head from processing the first to the j-th token. ℓ_g typically represents the cross-entropy loss function, and $\mathbb{1}_g(x_{i,j})$ is an indicator function, where $\mathbb{1}_g(x_{i,j}) = 1$ iff $role(x_{i,j}) = \mathsf{GEN}$; otherwise, it is 0. For tokens where $role(x_i) = \mathsf{RET}$, optimization employs $\mathcal{L}_r$:

$$\mathcal{L}_r = \sum_{i=1}^{|\mathcal{D}|}\sum_{j=1}^{|s_i|} \mathbb{1}_r(x_{i,j}) \cdot \ell_r(f_{\theta\backslash\pi_{\mathsf{Head}}}(x_{(i,\leq j)}), \\ f_{\theta\backslash\pi_{\mathsf{Head}}}(x_{(i,\leq j)^+}), f_{\theta\backslash\pi_{\mathsf{Head}}}(x_{(i,\leq j)^-})).$$

Here, ℓ_r is the contrastive loss (e.g., InfoNCE (van den Oord et al., 2018)), with $(i, \leq j)^-$ and $(i, \leq\ j)^+$ representing the sets of indices for negative and positive samples about sequence $x_{i,\leq j}$, respectively. The example in Figure 2 illustrates this concept clearly and effectively. $\mathbb{1}_r(x_{i,j})$ is an indicator function, where $\mathbb{1}_r(x_{i,j}) = 1$ iff $role(x_{i,j}) = \mathsf{RET}$; otherwise, it is 0. Finally, combining the two parts of loss by weighted addition gives the final optimization goal: $\mathcal{L} = \lambda_g\mathcal{L}_g + \lambda_r\mathcal{L}_r$, where λ_g and λ_r are hyperparameters. For a comprehensive overview of the detailed training procedures employed in other tasks, kindly refer to Figures 6 in Appendix.

Optimization. We use the standard Cross Entropy to optimize the loss function ℓ_g. For the loss function ℓ_r, prior work (Muennighoff et al., 2024) often utilizes the InfoNCE, which requires GradCache (Gao et al., 2021) to handle large batch sizes on low-memory GPUs, adding overhead and limiting to one positive sample per batch with a carefully chosen temperature hyperparameter. In contrast, we employ the hyperparameter-free BPR (Rendle et al., 2009), a pair-wise loss function. The ℓ_r loss is defined as: $\ell_r = -\log\ \sigma(\|f(x_{(i,\leq j)})\| \cdot (\|f(x_{(i,\leq j)^+_k})\| - \|f(x_{(i,\leq j)^-_k})\|))$, where $\sigma(\cdot)$ is the sigmoid function, $\|\cdot\|$ is normalization, and $(i, \leq j)^+_k$ and $(i, \leq j)^-_k$ are randomly selected from $(i, \leq j)^+$ and $(i, \leq j)^-$ respectively. Using BPR allows for Gradient Accumulation to support larger batch sizes and multiple positive samples per batch. Experimental results in Table 4 show that BPR reduces negative impacts on generative tasks and significantly enhances retrieval tasks.

3.3 Inference

For the standalone tasks of *Generation* and *Retrieval*, OneGen aligns with prior work. Here, we exclusively address the hybrid task of Generation and Retrieval. The inference process includes two primary steps: **1) Cache document embedding.** To facilitate efficient retrieval, we cache our document embedding after training is done, similar with prior work. Specifically, we append `[RD]` at the end of each document $x \in \mathcal{K}$ and use $f_{\theta\backslash\pi_{\mathsf{Head}}}$ to encoding them. We process all the documents in batch and the collection of these representations is stored in Emb_{doc} where $Emb_{doc} \in \mathbb{R}^{|\mathcal{K}|\times d}$. **2) Task generation.** Given an instruction, *2.1)* the LLM begins autoregressively generate the next token until $role(x_i) = \mathsf{RET}$ or $x_i \in$ `Terminator` (e.g., `</s>` in Llama2). If $role(x_i) = \mathsf{RET}$, then $f_{\theta\backslash\pi_{\mathsf{Head}}}(x_{\leq i})$ is used to retrieve from Emb_{doc} to obtain the set of most relevant documents $d_r \subset \mathcal{K}$. Here we use cosine similarity. *2.2) How to precede to generate the next token* x_{i+1}*?* Since $role(x_i) \neq \mathsf{GEN}$, the probability distribution $P(x_{i+1}|x_{\leq i})$ is not intrinsically modeled by LLM, thus, should be provided by user. For the multi-turn dialogue system, the user provides the token that initiates a new round of dialogue rather than the LLM. Specifically, for the $R \rightarrow G$ task, $P(x_{i+1}|x_{\leq i}) = d_r$ (i.e., directly concatenating the retrieved document). For $R \rightarrow G$ task, we specify $P(x_{i+1}|x_{\leq i}) =$ `[CON]`. *2.3)* Finally, the LLMs continue autoregressive prediction, repeating *step 2.1*. Detailed pseudocode and description can be found in Appendix F.3 and Appendix H.2.

Difference from Related Works. GritLM, GENRE (Cao et al., 2021b), and RICHES (Jain et al., 2024) can all perform specific hybrid retrieval and generation tasks using a single model. GritLM with causal attention first generates a query, then re-encodes it with bidirectional attention for retrieval in continuous space (i.e., using vector). GENRE and RICHES generate a query explicitly and use it to constrain decoding, enabling retrieval during generation in discrete space (i.e., using tokens) with just one forward pass. In contrast, OneGen performs retrieval in continuous space during generation, avoiding the need for two forward passes. Kindly refer to related work in Appendix A for details.

4 Experiments

4.1 Experimental Settings

We conduct extensive experiments across three different settings to validate OneGen's *effectiveness*, *efficiency in training and inference*, and *pluggabil-*

LLMs	Retriever			Dataset				AVG.
	Name	Dataset Name	Dataset Size	PopQA	TQA	Pub	ARC	
Toolformer (Schick et al., 2023)	Contriever	MS MARCO	1×10^6	-	48.8	-	-	-
Llama2$_{7B}$ (Touvron et al., 2023)	Contriever	MS MARCO	1×10^6	38.2	42.5	30.0	48.0	39.7
Alpaca$_{7B}$ (Dubois et al., 2023)	Contriever	MS MARCO	1×10^6	46.7	64.1	40.2	48.0	49.8
SAIL$_{7B}$ (Luo et al., 2023b)	Contriever	MS MARCO	1×10^6	-	-	69.2	48.4	-
Llama2-FT$_{7B}$ (Touvron et al., 2023)	Contriever	MS MARCO	1×10^6	48.7	57.3	64.3	65.8	59.0
Mistral$_{7B}$ (Jiang et al., 2023a)	Contriever	MS MARCO	1×10^6	23.2	49.3	52.0	39.0	40.9
GritLM$_{7B}$ (Muennighoff et al., 2024)	GritLM$_{7B}$	E5S(w/ TQA)	2×10^6	**58.0**	**66.5**	49.7	24.5	49.7
Self-RAG$_{7B}$ (Asai et al., 2024)	Contriever	MS MARCO	1×10^6	52.5	65.0	72.2	67.3	64.3
Self-RAG$_{7B}$ (+OneGen)	*Self*	*Sampled*	$\mathbf{6 \times 10^4}$	52.5	65.7	**75.1**	**70.1**	**65.8**

Table 2: Performance comparison across different datasets. "TQA" means TriviaQA, "Pub" means PublicHealth. The best and second-best results are indicated in bold and underlined. The complete table is shown in Table 9 of appendix. The details about Self-RAG are shown in appendix F.1.

ity. We train and evaluate a model for each setting independently. Specifically, the Single-hop QA involves one round of $R \rightarrow G$, Multi-hop QA entails multiple $R \rightarrow G$ executions, and Entity Linking involves multiple $G \rightarrow R$ executions. Due to page constraints, we provide a concise overview of the *workflow*, *baseline*, *training data*, *training backbones*, *evaluation datasets*, and *evaluation metrics* for each setting in order. Detailed information for each following settings can be found in the Appendix F.5, G.3, and H.4, respectively.

R → G Task: RAG for Single-hop QA. We apply OneGen directly to the Self-RAG (Asai et al., 2024) method (i.e., using Self-RAG as workflow), which incorporates adaptive retrieval and self-assessment. OneGen enhances it by enabling self-retrieval. The baselines we used are listed in Table 2. Llama2-7B-chat serves as the backbone of ours and Self-RAG. The training data is derived from the Self-RAG training dataset, modified according to § 3.2, comprising 150k instances. We construct positive and negative samples using heuristic rules for instances containing the [RQ] token. These samples are sourced from *wiki*, comprising 60k instances. We evaluate four datasets: PubHealth (Zhang et al., 2023a), ARC-Challenge (Clark et al., 2018), PopQA (Mallen et al., 2023), and TriviaQA (Joshi et al., 2017). For the first two, accuracy serves as the evaluation metric. For the others, evaluation is based on whether the model's output contained the ground truth. All evaluations are consistent with Self-RAG to ensure fair comparison.

R → G Task: RAG for Multi-hop QA. The following text sequence effectively demonstrates the workflow:" {Instruction} Are Alan Turing and Newton from the same country? What is Alan Turing's country? [RQ] {document 1} England. What is Newton's country? [RQ] {document 2} England. Therefore, yes.". This workflow combines CoT and RAG, where the gray tokens represent the role of CTX, the purple tokens represent RET, and the orange tokens represent GEN. The baseline uses the Contriever for retrieval, and the generator in the baseline is trained on the constructed data, with the optimization objective solely being $\mathcal{L}_g$. We sample 10% of the training dataset from HotpotQA and 2WIKI, using Qwen2-72B (Bai et al., 2023) for data construction. Heuristics rules are also employed to label the positive and negative samples. Evaluation is conducted using the validation set of HotpotQA and 2WIKI, as the ground truth of test sets is unavailable. We use EM and F1 to evaluate the model's generative capability and Recall@1 for its retrieval capability.

G → R Task: Entity Linking. The following text sequence effectively shows the workflow:" {Instruction} Steve Jobs founded Apple Inc. <LOC>Steve Jobs</LOC> [RQ] <CON> founded <LOC>Apple Inc</LOC> [RQ] <CON>.". The baselines we used are listed in Table 5. We employ the Wikipedia (totaling 6M documents, yet randomly sampling only 60K without careful selection) and AIDA (Hoffart et al., 2011) datasets, applying data augmentation to each sample in Wikipedia following established methods from previous studies (Cao et al., 2021a). We adopt heuristic rules to label each mention that includes an entity ID with both positive and negative documents, as detailed in the Appendix H.1. Llama2-7B-chat also serves as the backbone. We utilize the ELEVANT (Bast et al., 2022) for evaluation across seven datasets listed in Table 5, using Micro F1 to evaluate in-KB entities. The same datasets and metrics are applied to MD task. For ED task, following the ChatEL (Ding et al., 2024b), we evaluate nine datasets, maintaining the use of the Micro F1.

4.2 Main Results

In Table 2, Table 5, and Table 3, we report the performance of OneGen on three types of settings respectively, demonstrating that our method is both *effective*, *pluggable* and *training-efficient*.

R → G Task for Single-hop QA. From Table 2, we draw the following conclusions: *(1) OneGen demonstrates efficacy in $R \to G$ task, and joint training of retrieval and generation yields performance gains on the RAG task.* The Self-RAG endows LLMs with self-assessment and adaptive retrieval, while OneGen adds self-retrieval. Our method outperforms the original Self-RAG across all datasets, especially achieving improvements of 3.1pt on Pub dataset and 2.8pt on ARC dataset, validating the benefits of joint training, consistent with findings from RA-DIT (Lin et al., 2024) and GRIT (Muennighoff et al., 2024). However, ours on PopQA and TQA datasets remains inferior to GritLM-7B. We attribute this to using a larger retrieval dataset, E5S, which is twice the size of MS MARCO and 33 times larger than OneGen. Additionally, E5S includes TQA training data and higher-quality data. *(2) OneGen is highly efficient in training, with instruction-finetuned LLMs showing strong retrieval capabilities with minimal additional tuning.* It requires less and lower-quality retrieval data, achieving comparable performance with just 60K noisy samples and incomplete documents, without synthetic data. This efficiency aligns with findings from previous works like PromptEoL (Jiang et al., 2023c) and EchoEmbedding (Springer et al., 2024), which demonstrate excellent performance using prompt-based methods without further training.

R → G Task for Multi-hop QA. From Table 3, we additionally find that *OneGen remains effective across multiple $R \to G$ settings and works well with various models and scales.* It consistently outperforms the baseline on most datasets, backbones, and metrics. Notably, on the 2WIKI, OneGen achieves an average improvement of 5.2pt in EM and 4.6pt in F1. Additionally, our evaluation of end-to-end retrieval performance indicates that OneGen's performance on retrieval surpasses the baseline across all datasets and backbones.

G → R Task. From Table 5, we can draw the following conclusions: *(1) OneGen demonstrates effectiveness and strong generalization in the $G \to R$ task.* It outperforms ReFinED in the F1 score by 3.2pt on average and surpasses the most competitive baselines on the OKE16, REU, and K50 datasets by 3.1pt, 8.1pt, and 10.1pt, respectively. The lower performance on other datasets is attributed to insufficient training data, while the baselines utilized 6M data and we used only 1% of this amount. As INSGENEL (Xiao et al., 2023) has shown, increased training data can improve MD performance. In § 4.3.2, we analyze the bottlenecks in datasets with poorer outcomes, which lie in MD. Additionally, only 24% of entities in our candidate set participated in the training process. *(2) OneGen is highly efficient in training,* requiring merely 1% of data used in baselines.

4.3 Analysis

4.3.1 Efficiency at Inference Time

To assess the efficiency of OneGen during inference, we evaluate the inference time for various scenarios and tasks, as illustrated in Figure 3. For a fair comparison, all LLMs deployed are Mistral-v0.1-7B, operating with vLLM (Kwon et al., 2023) as the inference backend. Following the optimum-benchmark[1], all tokens are randomly generated and consistently numbered across baselines. Notably, the "Pipeline" in Figure 3(a-b) employ Contriever as the retriever, while the Figure 3(c), following the EntGPT, the retriever in Pipeline is Mistral-v0.1-7B, converting ED task into a QA task. The "Lower Bound" in Figure 3(a-b) denotes results obtained without retriever. Detailed settings are provided in the Appendix F.5.1 and H.4.1.

R → G Task. Figure 3(a-b) depict inference latency comparisons, with the default instruction in each round serving as the query. Figure 3(a) examines the influence of query length on inference speed in five rounds of $R \to G$. Figure 3(b) assesses how the retrieval frequency and context length affect inference speed with a fixed query length of 100 for each round. Key observations include: *1) Figure 3(a) shows that OneGen's inference process is efficient, with a notably greater increase in speed as query length extends, compared to Grit.* The efficiency stems from OneGen's use of extra retrieval tokens, maintaining overall time close to that of smaller models (e.g. Contriever) used as retrievers in the Pipeline. The improvement in inference speed with increasing query length varies from 4% to 20%, mainly due to OneGen's elimination of a second query forward pass com-

[1] https://github.com/huggingface/optimum-benchmark

		Generation Performance				Retrieval Performance	
		HotpotQA		2WIKI		HotpotQA	2WIKI
BackBone	**Retriever**	EM	F1	EM	F1	Recall@1	Recall@1
Llama2-7B	Contriever	52.83	65.64	70.02	74.35	73.76	68.75
	self	**54.82**	**67.93**	**75.02**	**78.86**	**75.90**	**69.79**
Llama3.1-7B	Contriever	53.72	66.46	70.92	75.29	69.79	66.80
	self	**55.38**	**68.35**	**75.88**	**79.60**	**72.55**	**68.98**
Qwen2-1.5B	Contriever	48.55	**61.02**	68.32	72.66	72.41	67.70
	self	**48.75**	60.98	**73.84**	**77.44**	**72.70**	**69.27**
Qwen2-7B	Contriever	53.32	66.22	70.80	74.86	74.15	69.01
	self	**55.12**	**67.60**	**76.17**	**79.82**	**75.68**	**69.96**

Table 3: In RAG for Multi-Hop QA settings, performance comparison across different datasets using different LLMs.

Task	**Loss Function** ($\mathcal{L}_r$)	
	InfoNCE	BPR
EL (7 datasets)	61.8	**64.0**
ED (9 datasets)	84.5	**86.5**
MD (7 datasets)	70.7	**71.5**

Table 4: Ablation study results of $\mathcal{L}_r$ on EL, ED, and Mention Detection (MD) tasks. The table reports average F1 scores for each task.

Method	**Cand. Size**	**Training Data**♦	**In-domain**	**Out-of-domain**						**AVG.**
			AIDA	OKE15	OKE16	REU	MSN	SPOT	K50	
Neural EL♦	< 30	AIDA	76.3	60.6	53.8	44.0	56.5	19.5	38.2	49.8
REL 2019◇	< 30	-	85.4	66.5	57.7	53.0	**77.8**	24.9	54.0	59.9
GENRE♦	< 30	WIKI 6M+AIDA	85.3	54.9	44.4	46.3	69.3	24.6	56.9	54.5
ReFinED♦	< 30	WIKI 6M+AIDA	**88.6**	**66.6**	61.2	49.8	74.7	22.2	62.8	60.8
Llama2$_{7B}$ (+OneGen)♦	1.25M	WIKI 60K+AIDA	83.1	63.5	**64.3**	**61.1**	74.2	**28.8**	**72.7**	**64.0**

Table 5: EL task performance on in-domain and out-of-domain test sets. The best value is in bold and the second best is underlined. The '♦' denotes end2end method, while the '◇' denotes pipelines.

pared with the alternatives (pipeline & GRIT). *2) Figure 3(b) reveals that OneGen maintains stable inference times, under extreme scenarios,* defined as one retrieval per dialogue round for 100 rounds with each round using 140 tokens and no semantic relevancy between rounds. Even when dialogue rounds reach 80, the context length consequently extends to 11K tokens, the increase in inference latency is minimal, ranging from 0.1% to 0.5%, showcasing the OneGen's stability. In practical applications, other methods involve query rewriting on context for retrieval, leading to substantial overhead. OneGen, which can operate without an explicit query, enhancing efficiency significantly.

G → R Task. Figure 3(c) demonstrates the effect of mention count on inference speed for EL tasks in sentences of 1K tokens, where each mention equates to a retrieval instance. Our findings include: *(1) OneGen is resource efficient,* deploying only one model compared to the dual-model setup in traditional pipelines. *(2) OneGen achieves enhanced inference efficiency, particularly as retrieval frequency increase.* In scenarios ranging from 10 to 100 retrievals, OneGen reduces inference time by 8% to 41%. This efficiency stems from bypassing the need to construct a query for each mention, unlike EntGPT. Furthermore, as the inference operates under the Next Token Prediction paradigm, it benefits from advanced large model serving techniques such as vLLM.

4.3.2 Impacts on Generation and Retrieval

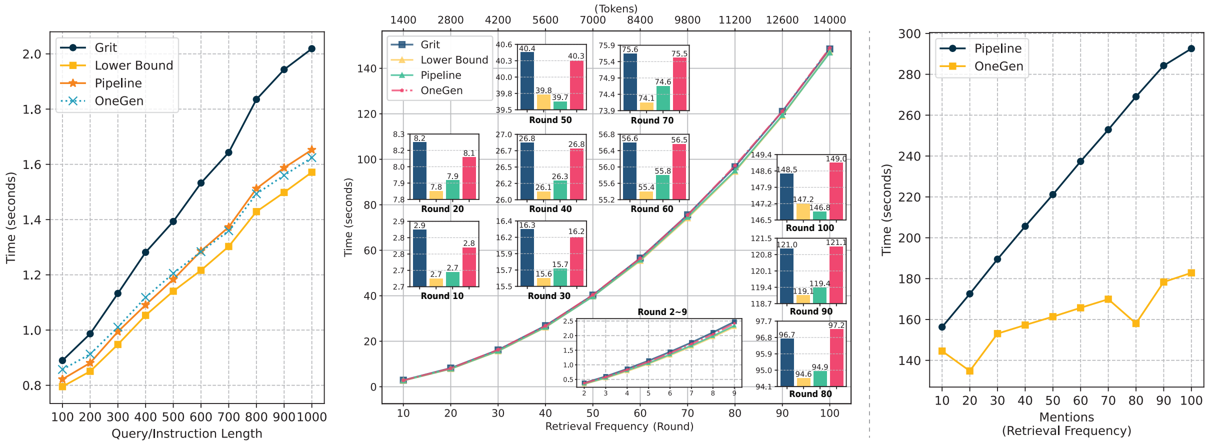
Here we examine whether situating retrieval and generation within the same context impacts the generative capacities of LLMs and assess the effectiveness of the retrieval. Given that § 4.2 evaluates end-to-end performance, where OneGen differs from other baselines in both the generation and retrieval modules, our evaluation principle here is to *fix the retriever to assess generative capabilities* and *fix the generator to evaluate retrieval capabilities.* More details are shown in Appendix F.5.2

G → R Task. *Same Retriever but different Generator:* Evaluation is performed through the MD task. Additionally, we train the LLM using the same data and hyperparameters with SFT (Supervised Fine-Tuning). Table 7 reports the average performance across seven datasets. Comparing the last two columns of Table 7, we find that **OneGen does not impair the LLM's generative capabilities.** *Same Generator but different Retriever:* We employ the Entity Disambiguation (ED) task to evaluate retrieval performance. Table 6 summarizes the average results over nine datasets, revealing that **OneGen significantly enhances the retrieval capacity of LLMs.**

4.3.3 Ablation Study

The more ablation studies, such as λ_r, implicit query, are shown in App. F.5.3, G.3.1, and H.4.2.

Loss Function. We examine the impact of $\mathcal{L}_r$,



(a) Runing Time Variation in RAG Task with Query Length.

(b) Runing Time Variation in RAG Task with Dialogue Turns for an extreme scenario (Fixed Query Length per Retrieval).

(c) Runing Time Variation in Entity Linking Task with Query Length.

Figure 3: Efficiency analysis of OneGen on RAG and Entity Linking tasks. All baselines maintain the same settings. For RAG, the output is 10 tokens, with a document length of 30 tokens. Figure (a) illustrates the impact of query length on RAG efficiency across five dialogue rounds. Figure (b) examines the influence of retrieval frequency and token length on RAG efficiency. Figure (c) depicts how retrieval frequency affects efficiency in Entity Linking tasks.

Method	ReFinED	ChatEL (GPT-4)	EntGPT-I (GPT-3.5)	OneGen (Llama2$_{7B}$)
AVG.	77.6	80.4	<u>84.3</u>	**86.5**

Table 6: Impact of OneGen on retrieval capabilities assessed through the ED task, presenting average F1 scores across nine datasets. Details are in Table 13 in Appendix.

Method	ReFinED	Llama2$_{7B}$ (SFT)	Llama2$_{7B}$ (OneGen)
AVG.	72.7	71.1	71.5

Table 7: Impact of OneGen on generation capabilities assessed through the Mention Detection task, presenting average F1 scores across seven datasets.

comparing BPR and InfoNCE, on EL, MD, and ED task. These tasks are assessed using seven, seven, and nine datasets respectively. Our results, presented in Table 4, indicate the BPR consistently surpasses InfoNCE in performance. This may be due to the overly restrictive of InfoNCE, which potentially limits the LLM's generative capabilities.

5 Conclusion and Future Work

In this paper, we utilize mathematical notation to formally unify generative tasks, retrieval task, and their composites such as RAG and EL. For composite tasks, we integrate retrieval and generation within the same context. Building upon this unified approach, we propose the OneGen training framework, which harmonizes and expands both generative and representative instruction tuning. We conduct extensive experiments on two distinct types of composite tasks, RAG and EL, to validate the pluggability, effectiveness, and efficiency of OneGen in training and inference. Furthermore, our results confirm that integrating generation and retrieval within the same context does not negatively impact the generative capabilities of LLMs, while also providing significant enhancements in retrieval capabilities. Future research directions include: 1) Extending OneGen to the multimodal domain for tasks such as multimodal RAG and multimodal EL. 2) Enhancing OneGen's training with diverse datasets to improve LLMs' capability for complex retrieval and generation tasks.

Limitations

Despite our comprehensive efforts, the study presents several limitations:

1) It remains unknown whether parameter-efficient fine-tuning methods such as LoRA (Hu et al., 2022) and QLoRA (Dettmers et al., 2023) could bring benefits for OneGen training. In this study, we utilized full-parameter fine-tuning. OneGen could potentially benefit from parameter-efficient fine-tuning, as recent work (Wang et al., 2024) has used LoRA to equip LLMs with retrieval.

2) The absence of performance evaluations in more diverse and extensive data scenarios. Although we achieved gains with limited data, a more diverse set of tasks and data, such as jointly training with Entity Linking, RAG, retrieval data, and generation data, might produce a model with enhanced capabilities.

Acknowledgments

We would like to express our sincere gratitude to the anonymous reviewers for their thoughtful and constructive feedback. We would like to thank Niels from HuggingFace for his valuable suggestions to improve the code. This work was supported by the National Natural Science Foundation of China (No. 62206246, No. NSFCU23B2055, No. NSFCU19B2027), the Fundamental Research Funds for the Central Universities (226-2023-00138), Zhejiang Provincial Natural Science Foundation of China (No. LGG22F030011), Yongjiang Talent Introduction Programme (2021A-156-G), and Information Technology Center and State Key Lab of CAD&CG, Zhejiang University. This work was supported by Ant Group and Zhejiang University - Ant Group Joint Laboratory of Knowledge Graph.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Tom Ayoola, Shubhi Tyagi, Joseph Fisher, Christos Christodoulopoulos, and Andrea Pierleoni. 2022. Refined: An efficient zero-shot-capable approach to end-to-end entity linking. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track, NAACL 2022, Hybrid: Seattle, Washington, USA + Online, July 10-15, 2022*, pages 209–220. Association for Computational Linguistics.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *CoRR*, abs/2309.16609.

Hannah Bast, Matthias Hertel, and Natalie Prange. 2022. ELEVANT: A fully automatic fine-grained entity linking evaluation and analysis tool. In *Proceedings of the The 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022 - System Demonstrations, Abu Dhabi, UAE, December 7-11, 2022*, pages 72–79. Association for Computational Linguistics.

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. 2024. Llm2vec: Large language models are secretly powerful text encoders. *CoRR*, abs/2404.05961.

Samuel Broscheit. 2019. Investigating entity knowledge in BERT with simple neural end-to-end entity linking. In *Proceedings of the 23rd Conference on Computational Natural Language Learning, CoNLL 2019, Hong Kong, China, November 3-4, 2019*, pages 677–685. Association for Computational Linguistics.

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: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021a. Highly parallel autoregressive entity linking with discriminative correction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 7662–7669. Association for Computational Linguistics.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021b. Autoregressive entity retrieval. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Chi-Min Chan, Chunpu Xu, Ruibin Yuan, Hongyin Luo, Wei Xue, Yike Guo, and Jie Fu. 2024. RQ-RAG: learning to refine queries for retrieval augmented generation. *CoRR*, abs/2404.00610.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*, pages 708–716. ACL.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

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 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Yifan Ding, Amrit Poudel, Qingkai Zeng, Tim Weninger, Balaji Veeramani, and Sanmitra Bhattacharya. 2024a. Entgpt: Linking generative large language models with knowledge bases. *CoRR*, abs/2402.06738.

Yifan Ding, Qingkai Zeng, and Tim Weninger. 2024b. Chatel: Entity linking with chatbots. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation, LREC/COLING 2024, 20-25 May, 2024, Torino, Italy*, pages 3086–3097. ELRA and ICCL.

Yann Dubois, Chen Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacafarm: A simulation framework for methods that learn from human feedback. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Luyu Gao, Yunyi Zhang, Jiawei Han, and Jamie Callan. 2021. Scaling deep contrastive learning batch size under memory limited setup. In *Proceedings of the 6th Workshop on Representation Learning for NLP, RepL4NLP@ACL-IJCNLP 2021, Online, August 6, 2021*, pages 316–321. Association for Computational Linguistics.

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.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing A multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 6609–6625. International Committee on Computational Linguistics.

Johannes Hoffart, Stephan Seufert, Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. 2012. KORE: keyphrase overlap relatedness for entity disambiguation. In *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, pages 545–554. ACM.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 782–792. ACL.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 541–550. The Association for Computer Linguistics.

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 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Trans. Mach. Learn. Res.*, 2022.

Palak Jain, Livio Baldini Soares, and Tom Kwiatkowski. 2024. From rag to riches: Retrieval interlaced with sequence generation. *Preprint*, arXiv:2407.00361.

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. 2023a. Mistral 7b. *CoRR*, abs/2310.06825.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023b. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13358–13376. Association for Computational Linguistics.

Ting Jiang, Shaohan Huang, Zhongzhi Luan, Deqing Wang, and Fuzhen Zhuang. 2023c. Scaling sentence embeddings with large language models. *CoRR*, abs/2307.16645.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023d. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 7969–7992. Association for Computational Linguistics.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1601–1611. Association for Computational Linguistics.

Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas Hofmann. 2018. End-to-end neural entity linking. In *Proceedings of the 22nd Conference on Computational Natural Language Learning, CoNLL 2018, Brussels, Belgium, October 31 - November 1, 2018*, pages 519–529. Association for Computational Linguistics.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Tuan Manh Lai, Heng Ji, and ChengXiang Zhai. 2022. Improving candidate retrieval with entity profile generation for wikidata entity linking. *Preprint*, arXiv:2202.13404.

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: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Rich James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, Luke Zettlemoyer, and Scott Yih. 2024. RA-DIT: retrieval-augmented dual instruction tuning. In *ICLR*. OpenReview.net.

Zihan Liu, Wei Ping, Rajarshi Roy, Peng Xu, Chankyu Lee, Mohammad Shoeybi, and Bryan Catanzaro. 2024. Chatqa: Surpassing gpt-4 on conversational qa and rag. *Preprint*, arXiv:2401.10225.

Haoran Luo, Haihong E, Zichen Tang, Shiyao Peng, Yikai Guo, Wentai Zhang, Chenghao Ma, Guanting Dong, Meina Song, and Wei Lin. 2023a. Chatkbqa: A generate-then-retrieve framework for knowledge base question answering with fine-tuned large language models. *CoRR*, abs/2310.08975.

Hongyin Luo, Yung-Sung Chuang, Yuan Gong, Tianhua Zhang, Yoon Kim, Xixin Wu, Danny Fox, Helen Meng, and James R. Glass. 2023b. SAIL: search-augmented instruction learning. *CoRR*, abs/2305.15225.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2024. Fine-tuning llama for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pages 2421–2425. ACM.

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 (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 9802–9822. Association for Computational Linguistics.

Shengyu Mao, Yong Jiang, Boli Chen, Xiao Li, Peng Wang, Xinyu Wang, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. 2024. Rafe: Ranking feedback improves query rewriting for RAG. *CoRR*, abs/2405.14431.

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2024. Generative representational instruction tuning. *CoRR*, abs/2402.09906.

Andrea Giovanni Nuzzolese, Anna Lisa Gentile, Valentina Presutti, Aldo Gangemi, Darío Garigliotti, and Roberto Navigli. 2015. Open knowledge extraction challenge. In *Semantic Web Evaluation Challenges - Second SemWebEval Challenge at ESWC 2015, Portorož, Slovenia, May 31 - June 4, 2015, Revised Selected Papers*, volume 548 of *Communications in Computer and Information Science*, pages 3–15. Springer.

Andrea Giovanni Nuzzolese, Anna Lisa Gentile, Valentina Presutti, Aldo Gangemi, Robert Meusel, and Heiko Paulheim. 2016. The second open knowledge extraction challenge. In *Semantic Web Challenges - Third SemWebEval Challenge at ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Revised Selected Papers*, volume 641 of *Communications in Computer and Information Science*, pages 3–16. Springer.

OpenAI. 2022. Chatgpt: Optimizing language models for dialogue. https://openai.com/blog/chatgpt/.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Chengwei Qin, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. 2023. Is chatgpt a general-purpose natural language processing task solver? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 1339–1384. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2020, Virtual Event / Atlanta, Georgia, USA, November 9-19, 2020*, page 20. IEEE/ACM.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: bayesian personalized ranking from implicit feedback. In *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*, pages 452–461. AUAI Press.

Michael Röder, Ricardo Usbeck, Sebastian Hellmann, Daniel Gerber, and Andreas Both. 2014. N^3 - A collection of datasets for named entity recognition and disambiguation in the NLP interchange format. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, May 26-31, 2014*, pages 3529–3533. European Language Resources Association (ELRA).

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: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Jacob Mitchell Springer, Suhas Kotha, Daniel Fried, Graham Neubig, and Aditi Raghunathan. 2024. Repetition improves language model embeddings. *CoRR*, abs/2402.15449.

Asa Cooper Stickland, Alexander Lyzhov, Jacob Pfau, Salsabila Mahdi, and Samuel R. Bowman. 2024. Steering without side effects: Improving post-deployment control of language models. *CoRR*, abs/2406.15518.

S. M. Towhidul Islam Tonmoy, S. M. Mehedi Zaman, Vinija Jain, Anku Rani, Vipula Rawte, Aman Chadha, and Amitava Das. 2024. A comprehensive survey of hallucination mitigation techniques in large language models. *CoRR*, abs/2401.01313.

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.

Alexander Matt Turner, Lisa Thiergart, David Udell, Gavin Leech, Ulisse Mini, and Monte MacDiarmid. 2023. Activation addition: Steering language models without optimization. *CoRR*, abs/2308.10248.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748.

Johannes M. van Hulst, Faegheh Hasibi, Koen Dercksen, Krisztian Balog, and Arjen P. de Vries. 2020. REL: an entity linker standing on the shoulders of giants. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 2197–2200. ACM.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Improving text embeddings with large language models. *CoRR*, abs/2401.00368.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6397–6407. Association for Computational Linguistics.

Zilin Xiao, Ming Gong, Jie Wu, Xingyao Zhang, Linjun Shou, and Daxin Jiang. 2023. Instructed language models with retrievers are powerful entity linkers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 2267–2282. Association for Computational Linguistics.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 10222–10240. Association for Computational Linguistics.

Yue Yu, Wei Ping, Zihan Liu, Boxin Wang, Jiaxuan You, Chao Zhang, Mohammad Shoeybi, and Bryan Catanzaro. 2024. Rankrag: Unifying context ranking with retrieval-augmented generation in llms. *CoRR*, abs/2407.02485.

Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Mingdao Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiao Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yifan An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhen Yang, Zhengxiao Du, Zhenyu Hou, and Zihan Wang. 2024. Chatglm: A family of large language models from GLM-130B to GLM-4 all tools. *CoRR*, abs/2406.12793.

Tianhua Zhang, Hongyin Luo, Yung-Sung Chuang, Wei Fang, Luc Gaitskell, Thomas Hartvigsen, Xixin Wu, Danny Fox, Helen Meng, and James R. Glass. 2023a. Interpretable unified language checking. *CoRR*, abs/2304.03728.

Wenzheng Zhang, Wenyue Hua, and Karl Stratos. 2022. Entqa: Entity linking as question answering. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. 2023b. Siren's song in the AI ocean: A survey on hallucination in large language models. *CoRR*, abs/2309.01219.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models. *CoRR*, abs/2303.18223.

Appendix

A Related Works **16**
A.1 LLM-based Retrieval 16
A.2 Composite Task 17

B Relation to prior work on LLM instruction tuning **18**

C Broader Application **18**

D Why does OneGen work? **18**

E OneGen's Features **19**

F R → G Task: RAG for Single-hop QA **20**
F.1 Introduction of Self-RAG 20
F.2 Training Details 20
F.3 Inference Details 21
F.4 Evaluation Details 21
F.5 Experiments 22
F.5.1 Efficiency Settings 22
F.5.2 Impacts on Generation and Retrieval 23
F.5.3 Ablation 24

G R → G Task: RAG for Multi-hop QA **25**
G.1 Training Details 25
G.2 Evaluation Details 26
G.3 Experiments 26
G.3.1 Ablation 26

H G → R Task: Entity Linking **29**
H.1 Training Details 29
H.2 Inference Details 30
H.3 Evaluation Details 30
H.4 Experiments 30
H.4.1 Efficiency Settings 30
H.4.2 Ablation 30

A Related Works

Natural Language Processing (NLP) tasks can be solved in two types of **spaces**: *continuous* and *discrete*. Encoder architectures (e.g., BERT (Devlin et al., 2019)) operate in the *continuous space*, where the basic units of processing are continuous *vectors*. Decoder architectures (e.g., GPT (Brown et al., 2020)) and Encoder-Decoder architectures (e.g., T5 (Raffel et al., 2020)) typically operate in the discrete space, processing discrete *tokens* as their fundamental units.

NLP **tasks** consist of two major categories: *Natural Language Understanding* (NLU) and *Natural Language Generation* (NLG). NLU tasks often involve classification (e.g., sentiment classification), while NLG tasks typically involve generation (e.g., novel generation). Specifically, classification tasks with a large number of classes are referred to as retrieval tasks. For clarity, we define tasks with fewer classes as classification tasks, and those with many classes as retrieval tasks. Generally, classification tasks can be reframed as generation tasks.

Figure 4 illustrates the relationship between NLP tasks and the spaces in which they are solved. As NLP has advanced, various composite tasks have emerged, such as Retrieval Augmented Generation (RAG) tasks and Entity Linking (EL), which often require coordination between retrieval and generation. These composite tasks can be divided into two categories based on whether retrieval serves generation: $R \rightarrow G$ tasks and $G \rightarrow R$ tasks. RAG is a representative $R \rightarrow G$ task, while EL is a $G \rightarrow R$ task, where the retrieved content does not directly serve generation.

For composite tasks, the generation task is always handled by a LLM. Therefore, we first introduce work related to LLM-based retrieval, and then we discuss related work on composite tasks.

A.1 LLM-based Retrieval

Numerous works have utilized LLMs to perform retrieval tasks. These can be categorized into continuous space retrieval and discrete space retrieval, depending on the unit used for retrieval. As shown in Figure 4, *discrete space retrieval cannot directly search from a vast candidate pool.* Therefore, it typically relies on recall methods to narrow down the search space. In contrast, *vector space retrieval can directly identify the target set from a large candidate pool.*

LLM-based Retrieval in Continuous Space. Continuous space is well-suited for retrieval tasks. The core of retrieval lies in how documents or queries are encoded. We classify the methods based on whether training is required and the type of attention mechanism used:

- **Prompt-based methods using Causal Attention.** PromptEOL (Jiang et al., 2023c) encodes documents by carefully crafting prompts and using logits from specific positions. EchoEmbedding (Springer et al., 2024) achieves document encoding by repeating the document and extracting logits from the final token or summing logits from corresponding positions. These methods demonstrate that current LLMs without training possess a certain degree of retrieval capability.

- **Trained methods using Causal Attention.** E5-Mistral (Wang et al., 2024) encodes documents by training on synthetic data, using the logits of the document's final token. EchoEmbedding (Springer et al., 2024) also provides a method for supervised training on logits of repeated documents.

- **Trained methods using Bidirectional Attention.** GritLM (Muennighoff et al., 2024) and LLM2VEC (BehnamGhader et al., 2024) replace the causal attention mechanism in LLMs with bidirectional attention and use Mean Pooling to obtain document encodings. GritLM is directly trained on supervised data, while LLM2VEC employs masked next-token prediction training with unsupervised data.

LLM-based Retrieval in Discrete Space. Discrete space retrieval is computationally intensive. It is commonly achieved through constrained decoding processes or structured as QA tasks:

- **Constrained decoding.** Approaches like GENRE (Cao et al., 2021b) and RICH (Jain et al., 2024) first generate a corresponding query with the LLM, which is then used to recall candidates from a pre-built index (such as a trie tree or FM-Index). The LLM then scores these candidates, guiding subsequent outputs to align with a specific document from the recalled set. Beam search is a main technique to implement the above process, where beam size is large and is often set to 10.

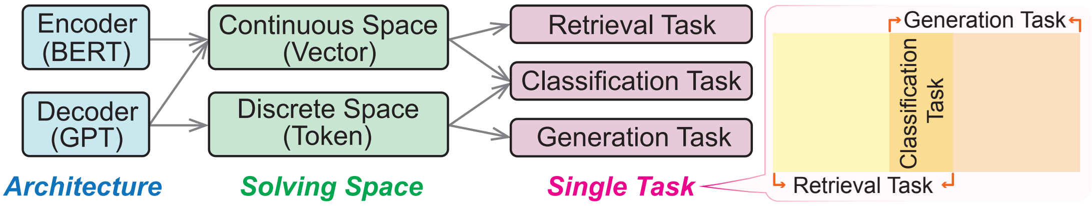


Figure 4: The relation between the model architecture, solving space, and tasks.

- **QA-based.** In these methods, the recalled candidate is typically concatenated directly into the text, instructing the LLM to output the index of the most relevant document or assign scores to each document, as seen in methods like RankRAG (Yu et al., 2024).

A.2 Composite Task

In this section, we define methods that require deploying two models during inference as *Pipeline* methods, regardless of whether these models are trained jointly or separately. Conversely, methods that rely on a single model are termed *Single Model* methods. It is important to note that if a method does not require the deployment of additional neural networks at the retrieval level and instead utilizes techniques such as BM25, TRIE trees, or FM-Index for retrieval, we classify these methods as belonging to the *Single Model* category.

Pipeline. The basic workflow of pipeline methods in Retrieval-Augmented Generation (RAG) is illustrated in Figure 5 (a). Typically, these methods involve deploying an extra model for retrieval followed by a large language model (LLM) for generation. This approach is prevalent in many works, including Self-RAG (Asai et al., 2024), ChatQA (Liu et al., 2024), RankRAG (Yu et al., 2024), and RQ-RAG (Chan et al., 2024). Since pipeline methods require the use of two separate models, query generation is essential. The LLM-generated query must be input into the retrieval model, necessitating two forward computations of the query. Specifically, Self-RAG introduces various scoring tokens to provide feedback on both the retrieved content and the generated output, optimizing retrieval augmentation. RQ-RAG enhances multi-step reasoning in question answering by decomposing the user's original query using Chain-of-Thought (CoT (Wei et al., 2022)) reasoning,. RankRAG improves retrieval performance by incorporating document ranking task during training, while ChatQA enhances the model's ability to understand contextual information, leading to better answer generation, as shown in Table 8.

Single Model. Single model methods significantly reduce memory overhead but lack a unified approach. As shown in Figure 4, we categorize existing single model methods into three types:

- **Encoder→Continuous Space.** Due to its encoder architecture, this approach cannot directly perform generation tasks. Entity Linking (EL) task involves Mention Detection (MD) and Entity Disambiguation (ED). ReFinED (Ayoola et al., 2022), although using two models during training—one for encoding documents and another for encoding queries and sequence labeling—aligns these encoders in representational space, so only the query encoder is required during inference. Thus the documents are encoded offline and cached. ReFinED handles Mention Detection as a sequence labeling task, effectively solving it as a classification problem. For Entity Disambiguation, ReFinED uses mean pooling on mentions and retrieves from a cached document embeddings. However, the encoder architecture's limitations restrict its application to other tasks.

- **Decoder→Discrete Space.** Given the Decoder architecture and the necessity of retrieval in a discrete space, the recall process becomes indispensable. Methods like GENRE (Cao et al., 2021b) and RICH (Jain et al., 2024) address this by constructing a TRIE tree and an FM-Index on the documents, respectively. These approaches perform retrieval concurrently with generation, as explained in Appendix A.1. However, the primary limitation for these approaches is time costing, often involving a beam search with a beam size of 10. Moreover, constructing training data for these approaches are stringent, requiring precise annotations for each retrieval point, unlike our method (see Appendix E).

- **Encoder→Continuous Space & Decoder→Discrete Space.** GritLM (Muennighoff et al., 2024) uses bidirectional attention to encode queries and documents during retrieval and causal attention during generation. This switching leads to two outcomes: 1) The query requires two forward computations since encoding and generation use different attention mechanisms, as shown in Figure 5(b). 2) The LLM's key-value cache cannot be effectively utilized.

B Relation to prior work on LLM instruction tuning

Recent work on representative instruction fine tuning (RIT) (Ma et al., 2024; Wang et al., 2024) demonstrated great potential for autoregressive LLMs in constructing high-quality embedding. They often use the logits (e.g. hidden states in the last layer) of EOS token appended at the end of the sentence to represent the input sequence. However, RIT models are only used for retrieval tasks due to the degenerated performance on generation after fine-tuning (Muennighoff et al., 2024). Grit (Muennighoff et al., 2024) uses different instruct prompts to control the switch between generation and embedding within a single LLM fine-tuned for both generation and representation instruction tuning and showed unifying the training task could improve the generation performance. The need for different prompts, however, makes it less efficient in applications requiring interleaved generation and retrieving. **From the perspective of training methods, OneGen is an extension of GIT and RIT, and it can degenerate to GIT and RIT.**

C Broader Application

Tasks that require both *generation* and *retrieval* (or *classification*) can be effectively handled using OneGen. Our approach enables the LLM to perform retrieval and classification during the generation process without any modification of architecture. Beyond the experimental tasks detailed in the main text, such as retrieval-augmented multi-hop reasoning and entity linking, here are two additional potential applications:

- **Text to Linked Knowledge Graph Triples.** Traditional knowledge graph construction typically involves several steps, including Named Entity Recognition (NER), Relation Extraction (RE), and Entity Linking (EL). With the advent of LLMs, it's possible to directly generate triples without linking from the sentence or document. Using OneGen, we can enhance this process by adding a special token (e.g., `[RQ]`) after each entity in the training data, allowing the LLM to generate linked knowledge graph triples in a single forward pass.

- **Controlled Generation Scenarios.** An example is KTS (Stickland et al., 2024), a method designed to prevent harmful outputs from LLM. Specifically, it classifies user inputs in advance. If the input is harmful, a pre-constructed steering vector (Turner et al., 2023) is inserted to ensure a safe response. Otherwise, the steering vector is not inserted. In implementation, by simply appending a special token (e.g., `[RQ]`) to each prompt, we can meet these requirements without relying on external classifiers or redundant encoding processes.

D Why does OneGen work?

From a training perspective, OneGen training approach is similar to that of multi-turn dialogue, where not all tokens contribute to the next token prediction loss. From an inference perspective, the multi-turn dialogue of LLM extends the new conversation by directly appending the user's next question after each dialogue round, whereas OneGen addresses this issue by appending a `<CON>` token. If we define the `[RQ]` token as the end of each dialogue round, the primary distinction between OneGentraining is that, in multi-turn dialogues, the last token of each round (such as `<|im_end|>` in Qwen2, `</s>` in Llama2, `<|eot_id|>` in Llama3) does not contribute to any loss during training and is thus redundant. In contrast, we leverage this token to encode the current intent. Therefore, since the multi-turn dialogue training method is effective, our approach is also effective.

So why is it possible to encode a query with just one token in the process of generation? First, LLMs inherently possess encoding capabilities, which can be enhanced with simple training to improve the LLM's retrieval abilities. Recent prompt-based methods, such as EchoEmbedding (Springer et al., 2024) and PromptEOL (Jiang et al., 2023c), have demonstrated that LLMs can achieve good retrieval results even without additional training. We believe

Method	Loss	Supported Data $(role(x_i) \in \{?\})$		
		{CTX, GEN}	{CTX, RET}	{CTX, RET, GEN}
GIT (SFT)	$\mathcal{L} = \mathcal{L}_g$	✓	✗	✗
RIT	$\mathcal{L} = \mathcal{L}_r$	✗	✓	✗
GRIT	$\mathcal{L} = \lambda_g \mathcal{L}_g + \lambda_r \mathcal{L}_r$	✓	✓	✗
OneGen	$\mathcal{L} = \lambda_g \mathcal{L}_g + \lambda_r \mathcal{L}_r$	✓	✓	✓

Table 8: Comparison of four Instruction Tuning

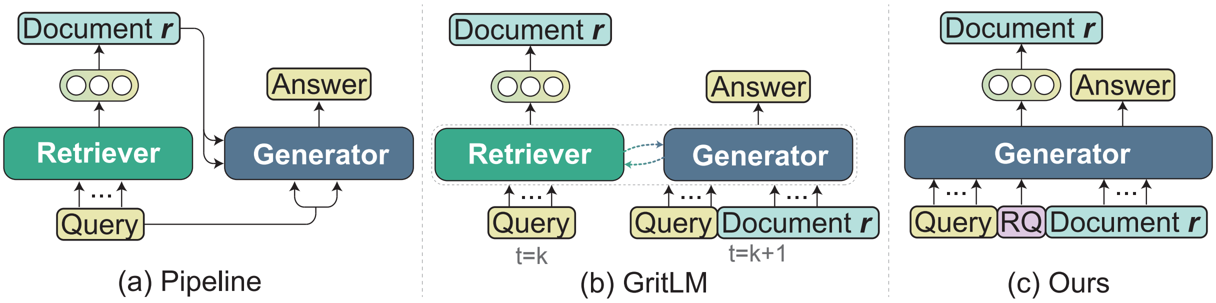


Figure 5: Comparison of three methods for completing RAG task.

that *an excellent painter (capability of generation) must have a great sense of aesthetics (capability of understanding), but the reverse is not necessarily true.* Second, many studies (Jiang et al., 2023b) have compressed prompts into a single token. Therefore, we can view special tokens, such as [RQ] and [RD], as compressed representations of the prompts "represent the current query" and "represent the current document".

E OneGen's Features

Support for Diverse Training Data. OneGen can handle various mixed data types, as shown in Table 8.

Pluggability. It is adaptable to different LLM architectures and sizes and can be integrated with existing methods such as Self-RAG (Asai et al., 2024) and RQ-RAG (Chan et al., 2024).

Training Efficiency. The use of the BPR (Rendle et al., 2009) loss function for optimizing $\mathcal{L}_r$ allows for Gradient Accumulation, which enables larger batch sizes for contrastive learning. Additionally, OneGen achieves competitive performance with less training data.

Inference Efficiency. OneGen avoids redundant computations by requiring only a single forward pass for queries, thereby reducing additional processing. Its unchanged model structure facilitates the use of existing inference acceleration techniques, such as vLLM (Kwon et al., 2023).

No Need for Query Rewriting or Query Generation. In the Multi-Hop QA setting, we demonstrate that the model can achieve comparable results to the baseline without generating or rewriting queries. Refer to Appendix G.3.1 for details.

Flexible Annotation Requirements. OneGen's retrieval operates in continuous space, meaning that annotations for retrieval components in training data do not need to be highly precise. For example, in the EL task, given an expected output "<LOC>Steve Jobs</LOC> [RQ] <CON> founded <LOC>Apple Inc</LOC> [RQ] <CON>", we may need to find the entities corresponding to"Steve Jobs" and "Apple Inc" in KB. However, for OneGen, it is sufficient to annotate only "Steve Jobs" or "Apple Inc", or even omit annotations altogether. The LLM can use annotations from other training data to optimize the [RQ] token.

F R → G Task: RAG for Single-hop QA

This section presents the details of the **R** → **G** task, including the detailed construction of the data, training details, inference details, and additional experimental results. Figure 5 shows a comparison of three methods for completing the RAG task.

F.1 Introduction of Self-RAG

Self-RAG (Asai et al., 2024) is a method designed for adaptive retrieval and self-assessment. Specifically, when a query input to the LLM, it begins generating responses in an autoregressive manner. When the LLM outputs the `[Retrieve]` token, it halts generation and invokes a retriever (e.g., Contriever (Izacard et al., 2022)) to retrieve relevant documents. These documents are then appended to the current context using the template "`{history}<paragraph>relevant document</paragraph>`". The LLM is then required to output either `[Relevant]` or `[Irrelevant]` to determine if the retrieved documents are relevant to the query. If the documents are deemed irrelevant, the LLM outputs the query again and performs another retrieval. If relevant, the LLM continues to generate the answer. Self-RAG employs tokens such as `[Fully Supported]`, `[Partially Supported]`, and `[No Support]` to evaluate whether the generated answer aligns with the retrieved documents, and uses rating tokens like `[Utility:5]`, `[Utility:4]`, `[Utility:3]`, `[Utility:2]`, and `[Utility:1]` to assess the relevance of the answer to the original question. For more details, we refer readers to the original paper.

F.2 Training Details

A comprehensive example of training Self-RAG is illustrated in Figure 6 (b).

Data Reconstruction. Here, we outline the modifications we implemented in the training dataset for Self-RAG (Asai et al., 2024). The following diagram presents a typical Self-RAG training example, featuring special tokens highlighted in blue and red. During training, all tokens except for the black text are involved in computing the generative loss.

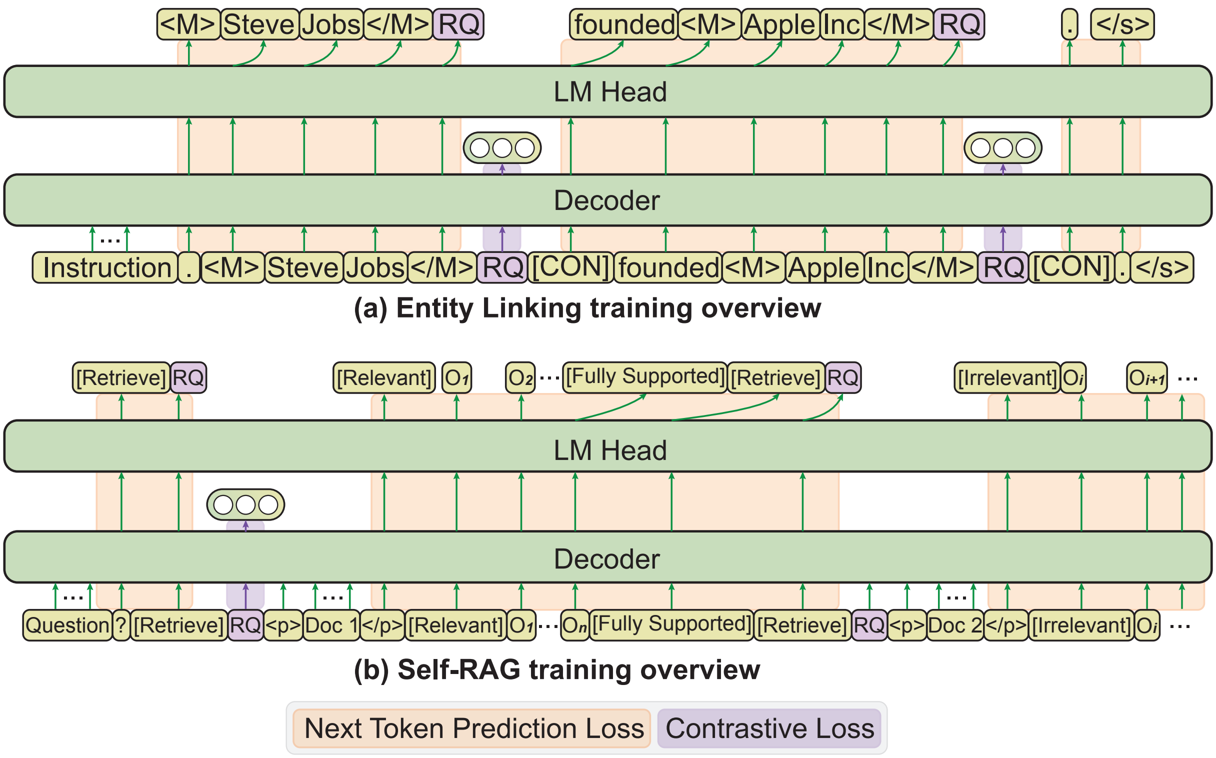
> **Input:**
> Q: Where do you work?
> A: Apple Inc., the iPhone is one of their products.
> Q: Who is the CEO?
> **Output:**
> `[Retrieval]` `<p>` Timothy Cook (born November 1, 1960) is an American business executive who is the current chief executive officer of Apple Inc. `</p>` `[Relevant]` Tim Cook `[Fully Supported]` `[Utility:5]`.

Take this for example, we introduce a `[RQ]` token immediately following the **[Retrieval]** token (illustrated as the purple token in the following diagram), enabling the LLM to extract the semantic information of the query at this point, with the rest of the setup remaining unchanged. In training, on top of the original setting, we compute a representative loss specifically for the purple token, e.g.,$\mathit{role}(\texttt{[RQ]}) = \texttt{RET}$. It should be noted that a `[RQ]` token is appended after every `[Retrieval]` token. However, not all are included in the training process. Specifically, if a sequence formatted as `[Retrieval][RQ]<p>`···`</p>[Irrelevant]` occurs, indicating that the document encapsulated within `<p>`···`</p>` is followed by a `[Irrelevant]` token, then the `[RQ]` token in this context does not contribute to the calculation of loss.

> **Input:**
> Q: Where do you work?
> A: Apple Inc., the iPhone is one of their products.
> Q: Who is the CEO?
> **Output:**
> `[Retrieval]` **`[RQ]`** `<p>` Timothy Cook (born November 1, 1960) is an American business executive who is the current chief executive officer of Apple Inc. `</p>` `[Relevant]` Tim Cook `[Fully Supported]` `[Utility:5]`.

The calculation of the representative loss requires positive and negative samples. Hence, for the positive samples, we use the content enclosed within `<p>`...`</p>` tags. For the negative samples, we utilize the Mistral-E5 (Wang et al., 2024) to embed the document corpus [2] and select documents

[2] https://dl.fbaipublicfiles.com/dpr/wikipedia_split/psgs_w100.tsv.gz


<M> Steve Jobs </M> RQ
founded <M> Apple Inc </M> RQ
. </s>
LM Head
Decoder
Instruction . <M> Steve Jobs </M> RQ [CON] founded <M> Apple Inc </M> RQ [CON] . </s>
[Retrieve] RQ
[Relevant] O₁
O₂ ··· [Fully Supported] [Retrieve] RQ
[Irrelevant] Oᵢ
Oᵢ₊₁ ···
LM Head
Decoder
Question ? [Retrieve] RQ <p> Doc 1 </p> [Relevant] O₁ ··· Oₙ [Fully Supported] [Retrieve] RQ <p> Doc 2 </p> [Irrelevant] Oᵢ ···
Next Token Prediction Loss
Contrastive Loss

(a) Entity Linking training overview

(b) Self-RAG training overview

Figure 6: The detailed training process of Entity Linking and Self-RAG.

ranked from 5985 to 6000 as negative samples. The numbers 5985 and 6000 are hyperparameters for constructing negative samples. Since the Self-RAG training data does not include negative samples (as it uses the Contriever, which doesn't require training), and OneGen needs to train retrieval capabilities, we have to create negative samples. In the document pool provided by Self-RAG, documents are divided into different chunks, and multiple chunks or documents may correspond to the answer for a given query. This means that not only one document necessarily contains the correct answer. To avoid selecting false negatives, we decided to use chunks ranked between 5985 and 6000 as negative samples. Examples of both positive and negative samples are provided below.

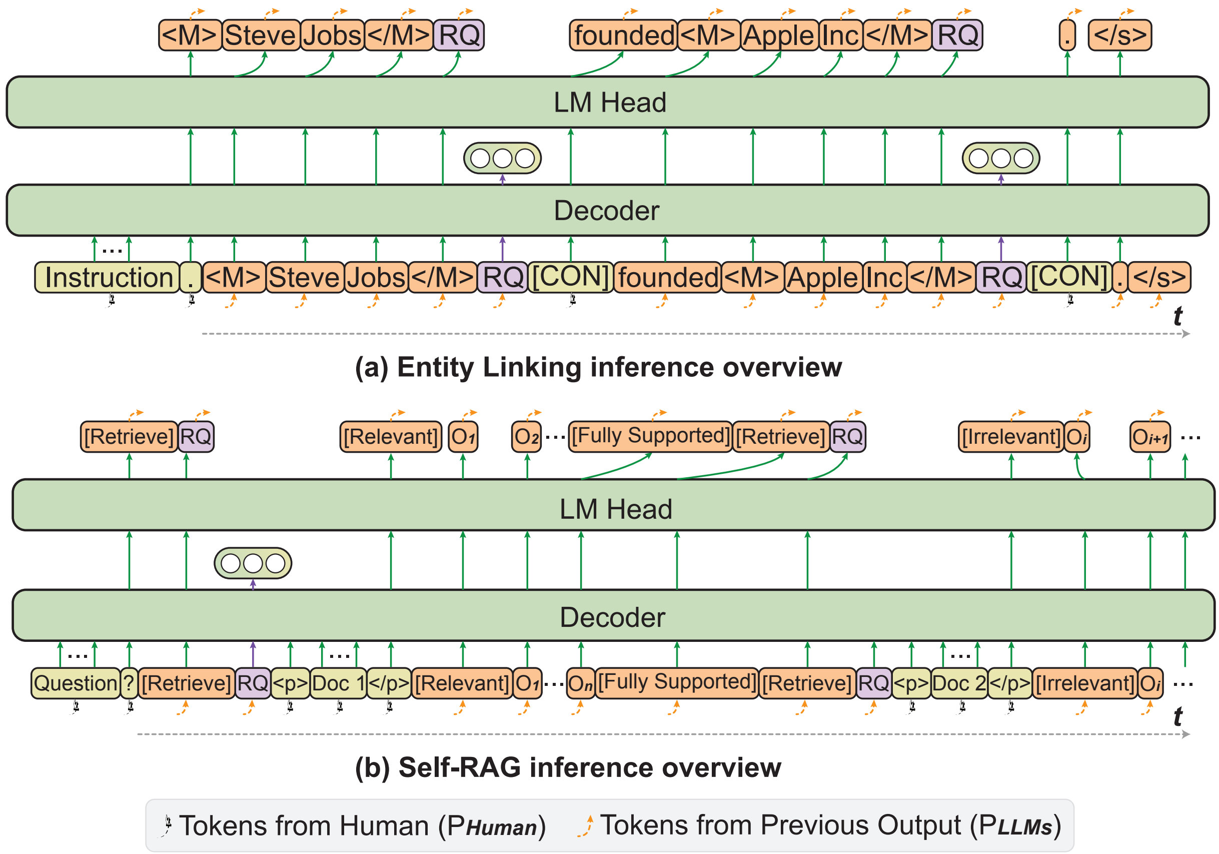
Implementation Details. The original dataset for Self-RAG (Asai et al., 2024) training comprises 150k instances, of which 60k are suitable for computing the representative loss. We perform comprehensive training on eight A800 machines utilizing the DeepSpeed ZeRO-3 (Rajbhandari et al., 2020) strategy for memory efficiency. We set the gradient accumulation to 4, and the batch size per GPU is 3, resulting in a final global batch size of $8 \times 3 \times 4 = 96$. Training is conducted over 3 epochs with a learning rate set at 2e-5 and a 3% warm-up period. Both λ_g and λ_r are set at 1. Data is sampled randomly. Within a batch, the loss $\mathcal{L}_r$ is computed if the `[RQ]` token is present. Otherwise, it is omitted. For each `[RQ]` token, we sample one positive and two negative documents, with negative documents shared across the batch.

F.3 Inference Details

Algorithm F.3 presents the pseudocode for Self-RAG inference. Figure 7(b) provides a schematic representation of the Self-RAG inference process.

F.4 Evaluation Details

Baselines. Baselines can be categorized into three types. The first, named *LLMs with proprietary data*, involves directly questions into strong models like ChatGPT, without using any retrieval documents or specialized training. The second named *Baselines without retrieval* uses LLMs such as Llama2 and Alpaca in 7B-Chat and 13B-Chat configurations. The third named *Baselines with retrieval*. Except for Llama2-7B-FT (Touvron et al., 2023), which is finetuned using Self-RAG training data without reflection token, responses are generated from the concatenation of questions and retrieved documents All the baselines use Contriever-MSMARCO (Izacard et al., 2022) as the retriever, except for the GritLM-7B, which serves as its own retriever.



<M> Steve Jobs </M> RQ founded <M> Apple Inc </M> RQ . </s>
LM Head
Decoder
Instruction . <M> Steve Jobs </M> RQ [CON] founded <M> Apple Inc </M> RQ [CON] . </s>
t

(a) Entity Linking inference overview

[Retrieve] RQ [Relevant] O1 O2 … [Fully Supported] [Retrieve] RQ [Irrelevant] Oi Oi+1 …
LM Head
Decoder
Question ? [Retrieve] RQ <p> Doc 1 </p> [Relevant] O1 … On [Fully Supported] [Retrieve] RQ <p> Doc 2 </p> [Irrelevant] Oi …
t

(b) Self-RAG inference overview

Tokens from Human (PHuman)
Tokens from Previous Output (PLLMs)

Figure 7: The detailed inference process of Entity Linking and Self-RAG.

Positive Example:
Timothy Cook (born November 1, 1960) is an American business executive who is the current chief executive officer of Apple Inc.**[RD]**
Negative Example:
After graduating from Auburn University, Cook spent twelve years in IBM's personal computer business, ultimately as director of North American fulfillment.**[RD]**

Figure 8: A case for data reconstruction of positive document and negative document about Self-RAG. Add [RD] token at the end of each document.

Evaluation Setup, Datasets, and Metric. We evaluated OneGen on four datasets: PubHealth (Zhang et al., 2023a), ARC-Challenge (Clark et al., 2018), PopQA (Mallen et al., 2023) and TriviaQA (Joshi et al., 2017). For the first two, we use accuracy as an evaluation metric. For the others, we assess performance by checking if the model generations contain gold answers, rather than insisting on exact matches, as per the method used by (Mallen et al., 2023); (Asai et al., 2024); (Schick et al., 2023). Moreover, each candidate document corresponding to a question is provided by its respective dataset.

F.5 Experiments

F.5.1 Efficiency Settings

We conduct tests on a single 40GB A100 card, with an `Intel(R) Xeon(R) Platinum 8352V CPU @ 2.10GHz`. We test RAG in a multi-turn dialogue setting, where retrieval is required at each interaction. Initially, the LLM is provided with an instruction comprising 100 tokens. The LLM first generates a token to determine the necessity of retrieval. After that, a query formulated from the same 100 tokens is sent to the retriever, which retrieves a document containing 30 tokens. This document is subsequently concatenated to the initial instruction, and the LLM generates an output of 10 tokens as the response. Batch size is set to 1 during the test. Prior to actual testing, the model undergoes 10 warm-up cycles. We assume mandatory retrieval in each iteration and that the instruction and query are of equivalent length. However, it is notable that most existing methodologies do not circumvent the

Algorithm 1 RAG Inference

```
Input:
    LLM trained with OneGen, denoted as $\hat{f}(\cdot)$
    LLM without the LM-Head, denoted as $f(\cdot)$
    Pre-cached document vector library $Emb_{doc}$
    Instruction $x$
    Cosine similarity computation function CosineSimilarity()
    Function to sort and return the corresponding documents Top1Doc()
Output:
    Answer $History$

$History \leftarrow x$
$NextToken \leftarrow \hat{f}(History)$
while $NextToken \notin$ Terminator do
    $History \leftarrow History \cup \{NextToken\}$
    if $role(NextToken) =$ RET then                                   ▷ Retrieval on demand
        $scores \leftarrow$ CosineSimilarity($f(History), Emb_{doc}$)
        RelevantDoc $\leftarrow$ Top1Doc($scores$)
        $History \leftarrow History \cup \{$RelevantDoc$\}$
    end if
    $NextToken \leftarrow \hat{f}(History)$                                  ▷ Generation
end while
return $History$
```

LLMs	Retriever			Dataset				AVG.
	Name	Dataset Name	Dataset Size	PopQA	TQA	Pub	ARC	
			LLMs with proprierary data					
Llama2-c_{13B}	-	-	-	20.0	59.3	49.4	38.4	41.8
Ret-Llama2-c_{13B}	-	-	-	51.8	59.8	52.1	37.9	50.4
ChatGPT	-	-	-	29.3	74.3	70.1	75.3	62.3
Ret-ChatGPT	-	-	-	50.8	65.7	54.7	75.3	61.6
			Baselines without retrieval					
Llama2$_{7B}$ (Touvron et al., 2023)	-	-	-	14.7	30.5	34.2	21.8	25.3
Alpaca$_{7B}$ (Dubois et al., 2023)	-	-	-	23.6	54.5	49.8	45.0	43.2
Llama2$_{13B}$ (Touvron et al., 2023)	-	-	-	14.7	38.5	29.4	29.4	28.0
Alpaca$_{13B}$ (Dubois et al., 2023)	-	-	-	24.4	61.3	55.5	54.9	49.0
			Baselines with retrieval					
Toolformer (Schick et al., 2023)	Contriever	MS MARCO	1×10^6	-	48.8	-	-	-
Llama2$_{7B}$ (Touvron et al., 2023)	Contriever	MS MARCO	1×10^6	38.2	42.5	30.0	48.0	39.7
Alpaca$_{7B}$ (Dubois et al., 2023)	Contriever	MS MARCO	1×10^6	46.7	64.1	40.2	48.0	49.8
SAIL$_{7B}$ (Luo et al., 2023b)	Contriever	MS MARCO	1×10^6	-	-	69.2	48.4	-
Llama2-FT$_{7B}$ (Touvron et al., 2023)	Contriever	MS MARCO	1×10^6	48.7	57.3	64.3	65.8	59.0
Mistral$_{7B}$ (Jiang et al., 2023a)	Contriever	MS MARCO	1×10^6	23.2	49.3	52.0	39.0	40.9
GritLM$_{7B}$ (Muennighoff et al., 2024)	GritLM$_{7B}$	E5S(w/ TQA)	2×10^6	**58.0**	**66.5**	49.7	24.5	49.7
Self-RAG$_{7B}$ (Asai et al., 2024)	Contriever	MS MARCO	1×10^6	$\underline{52.5}$	65.0	$\underline{72.2}$	$\underline{67.3}$	$\underline{64.3}$
Self-RAG$_{7B}$ (+OneGen)	*Self*	*Sampled*	$\mathbf{6 \times 10^4}$	$\underline{52.5}$	$\underline{65.7}$	**75.1**	**70.1**	**65.8**

Table 9: Performance comparison across different datasets. Best and second-best results within 'Baselines with retrieval' are indicated in bold and underlined, respectively.

step of query rewriting. In Figure 3(a), we conduct five dialogue rounds and report the cumulative time expended across these dialogues. For OneGen, an additional token is generated in each round of dialogue for retrieval purposes.

F.5.2 Impacts on Generation and Retrieval

R → G Task. *Same Retriever but different Generator:* We employ Contriever as the retriever. Specifically, the document ranks are obtained from it. Comparing the red and blue bars in Figure 9, we observe a slight decrease in performance of approximately 0.7 points on PopQA and a decrease of 2.0

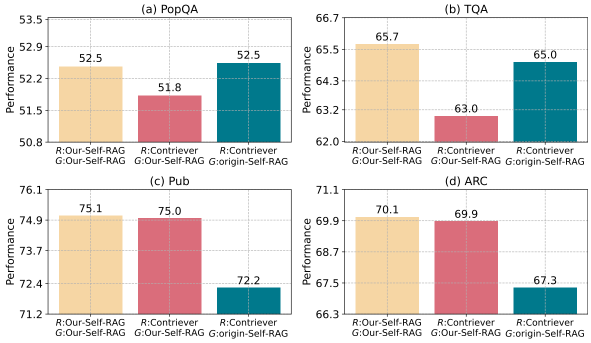


Figure 9: Performance comparison of three configurations across four datasets in a RAG task. Each bar represents a specific setup, with 'R' denoting the model used for retrieval and 'G' indicating the model used for generation.

points on TriviaQA. Conversely, performance improved by 2.8 and 2.6 points on the Pub and ARC datasets respectively, likely due to joint training, consistent with findings from RA-DIT (Lin et al., 2024) and Grit (Muennighoff et al., 2024). **Overall, OneGen does not adversely affect the generative capabilities of LLMs.** *Same Generator but different Retriever:* We assess the performance of different retrievers by examining the output of the generator. Observing the yellow and red bars in Figure 9, we note an increase in performance of 0.7 and 2.7 points on the PopQA and TriviaQA datasets, respectively. Slight gains were also observed in the other datasets, indicating that **OneGen effectively enhances the retrieval capabilities of LLMs.**

F.5.3 Ablation

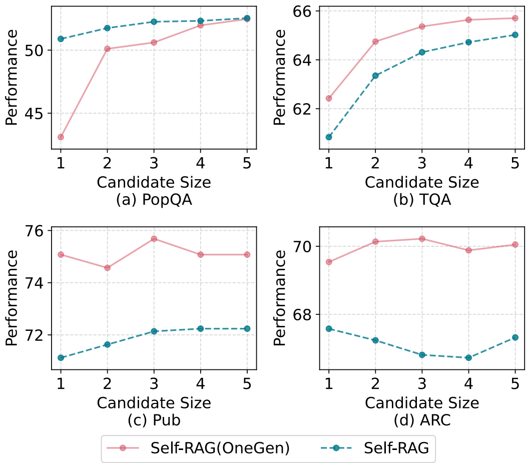


Figure 10: Performance of Self-RAG on four datasets at different candidate sizes from Top 1 to 5.

Top-K. Self-RAG is capable of autonomously evaluating its retrieval outcomes. Specifically, if a retriever produces the Top-K results, Self-RAG processes and assesses each result independently, necessitating K evaluations. Based on these assessments, it selects the highest-scoring result. We explore values of K ranging from 1 to 5, with the findings depicted in Figure 10. The observed trend is consistent between Self-RAG and Self-RAG (OneGen). Performance on the PopQA and TriviaQA datasets increases with larger K, demonstrating the effectiveness of the self-assessment mechanism.

G R → G Task: RAG for Multi-hop QA

G.1 Training Details

Data Reconstruction. Our data reconstruction process consists of two stages. First, we use Qwen2-72B with our designed prompts illustrated in Figure 13 to further annotate the data from the HotpotQA (Yang et al., 2018) and 2WIKI (Ho et al., 2020) datasets. Specifically, each query is decomposed into a sequence of sub-queries, and each sub-query is matched with the corresponding documents. After annotation, a modified example is as follows:

Input:
<Instruction>. Is Yangzhong or Cenxi located in the east of Guangxi, People's Republic of China?

Output:
First, In which province is Yangzhong located? [RQ]
<paragraph>wiki title: Yangzhong
content: Yangzhong is a beautiful city. Yangzhong is also a county-level city under the administration of Zhenjiang, Jiangsu province, China. It is the easternmost county-level division of Zhenjiang City.</paragraph>
Yangzhong is located in Jiangsu province, China.

Second, Is Cenxi located in the east of Guangxi, People's Republic of China? [RQ]
<paragraph>wiki title: Cenxi
content: Cenxi is a county-level city under the administration of Wuzhou City, in the east of Guangxi, People's Republic of China.</paragraph>
Yes, Cenxi is located in the east of Guangxi, People's Republic of China.

Therefore, Cenxi is the city located in the east of Guangxi, People's Republic of China, not Yangzhong.
<FINAL-ANSWER>Cenxi</FINAL-ANSWER>

In this example, the tokens highlighted in brown are involved in the calculation of the $\mathcal{L}_g$ loss, while those highlighted in purple are involved in the calculation of the $\mathcal{L}_r$ loss. Special tokens used include <paragraph>, </paragraph>, and [RQ].

Since both 2WIKI and HotpotQA provide candidate documents for each query and each document is segmented into sentences, our strategy for document representation is to perform a forward pass on each document. If a document contains n sentences, this results in n representations. Specifically, we append the [RD] token to each sentence within the document. For illustration, consider the above query "In which province is Yangzhong located?" to explain how we select positive and negative samples. Suppose we have two documents:

Doucment 1:
Yangzhong is a beautiful city. [RD]$_1$ Yangzhong is also a county-level city under the administration of Zhenjiang, Jiangsu province, China. [RD]$_2$ It is the easternmost county-level division of Zhenjiang City. [RD]$_3$

Doucment 2:
Cenxi is a county-level city under the administration of Wuzhou City, in the east of Guangxi, People's Republic of China.[RD]$_4$

Figure 11: A case for data reconstruction of positive document and negative document about Multi-hop QA. Add [RD] token at the end of each sentence in every document.

To distinguish them, we append an ID to each [RD] token, though in practice they are identical. The positive samples for this query are [RD]$_2$ and [RD]$_3$, while the negative samples are [RD]$_1$ and [RD]$_4$. A segment containing the expected answer is considered a positive sample; otherwise, it is a negative sample. Thus, [RD]$_2$ is expected to retrieve information from [RD]$_1$, and [RD]$_3$ is expected to capture representations from the preceding two sentences.

Implementation Details. Following the previous steps, we randomly sampled 10% of the HotpotQA and 2WIKI training datasets for annotation. After the annotation process, we obtained 9,044 samples from HotpotQA and 16,745 samples from 2Wiki, resulting in a total of 25,789 training samples. For evaluation, we utilize the complete validation sets of HotpotQA and 2WIKI, as the original

BackBone	Hyper Parameters		
	Pos. per Sent.	Neg. per Pos.	Max Length
Llama2-7B	2	6	1200
Llama3.1-7B	2	2	1200
Qwen2-1.5B	2	4	1100
Qwen2-7B	2	2	1100

Table 10: Training hyper parameters for different backbone in Multi-hop QA setting.

test sets do not provide ground truth labels. The HotpotQA validation set consists of 7,405 samples, while the 2WIKI validation set comprises 12,576 samples. We perform comprehensive training on eight A800 machines utilizing the DeepSpeed ZeRO-3 strategy for memory efficiency. We set the gradient accumulation to 4, and the batch size per GPU is 2, resulting in a final global batch size of $8 \times 2 \times 4 = 64$. Training is conducted over 3 epochs with a learning rate set at 2e-5 and a 3% warm-up period. Both λ_g and λ_r are set at 1. Other hyper-parameters are shown in Table 10.

G.2 Evaluation Details

Baselines. We use a pipeline method as the baseline for the Multi-hop QA setting, where the pipeline alternates between the retriever and the generator. For the retriever, we employ the untrained Contriever (Izacard et al., 2022), which is consistent with the retrievers used in RQ-RAG (Chan et al., 2024) and Self-RAG (Asai et al., 2024). For the generator, we train on data constructed as described in Appendix G.1. The only difference from OneGen is the omission of the $\mathcal{L}_r$ loss, meaning that the `[RQ]` token used as input to the LLM is not involved in optimization. During the inference, the generation of the `[RQ]` token by the LLM indicates the need to call the retriever. We input each generated sub-query into the retriever for retrieval.

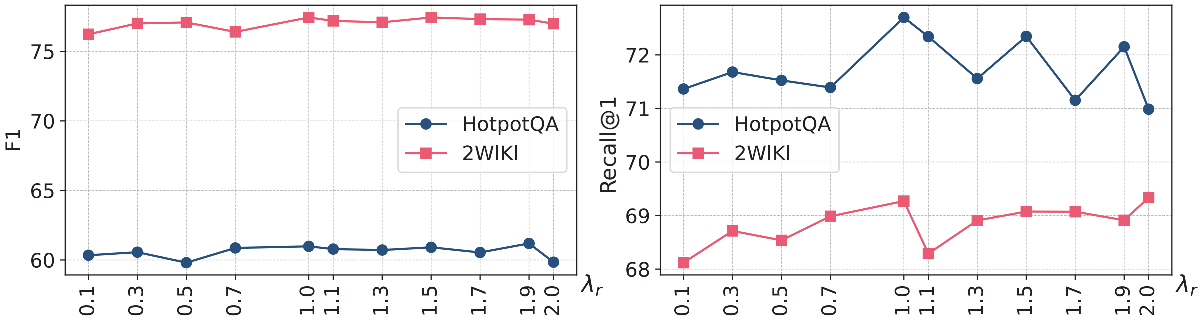
Evaluation Metrics. We utilize the code provided by the original papers (Yang et al., 2018; Ho et al., 2020) to evaluate the generation of both the pipeline and our method using F1 and EM metrics. For retrieval evaluation, since the LLM used by the pipeline and the LLM trained with OneGen have different model parameters, the queries generated for the same question differ. Thus, we report the retrieval results for queries generated by Contriever from the pipeline, and the retrieval results for queries generated using OneGen. Specifically, for a given query requiring two steps of reasoning, two sub-queries are generated, each retrieving a relevant document. If these retrieved documents match the ground truth, the retrieval is considered correct; otherwise, it is considered incorrect.

G.3 Experiments

G.3.1 Ablation

Hyper parameter λ_r. We examine the impact of λ_r using the Qwen2-1.5B for Multi-hop QA task. We search the λ_r from $\{0.1, 0.3, 0.7, 1.0, 1.3, 1.5, 1.7, 1.9, 2.0\}$ and the result are presented in Figure 12. We find that OneGen is insensitive to hyper- parameters λ_r, demonstrating the robustness of the OneGen.

Implicit Query. Here, we remove the explicit query to observe the performance. Take the previous query "`In which province is Yangzhong located?`" in the of *Data Reconstruction* Appendix G.1 as an example, we replace the explicit query with the implicit query "`The question is:`". Table 11 shows the result. We find that OneGen can get a good performance under the implicit query settings.



(a) End-to-End Generation.

(b) End-to-End Retrieval.

Figure 12: Ablation study for λ_r using Qwen2-1.5B.

Method	**HotpotQA**		**2WIKI**	
	EM	F1	EM	F1
Llama2-7B + Explicit Query + OneGen	**54.82**	**67.93**	**75.02**	**78.86**
Llama2-7B + Implicit Query + OneGen	51.02	63.27	**72.92**	**79.23**
Llama2-7B + Contriever	52.83	65.64	70.02	74.35

Table 11: Ablation study for using the implicit query.

You excel in question decomposition. Starting with a question, related documents, and the final answer corresponding to the question, your task is to break down the question into sub-questions. Each sub-question should be connected to relevant documents, leading to the generation of answers for each sub-question. It's important to consider potential connections between these sub-questions.
Here's an example input:
<QUESTION>
Are both magazines, the Woman's Viewpoint and Pick Me Up, British publications?
</QUESTION>
<ANSWER> no </ANSWER>
<RELATED-DOC-1>
wiki title: Pick Me Up (magazine)
content: Pick Me Up! is a British weekly women's magazine that is published through the IPC Media group.
</RELATED-DOC-1>
<RELATED-DOC-2>
wiki title: Woman's Viewpoint (magazine) content: The Woman's Viewpoint was a woman's magazine founded in Texas in 1923 and published by Florence M. Sterling. The magazine was progressive and ran from 1923 to 1927.
</RELATED-DOC-2>

Here's an example output for the given input:
<SUB-QUESTION-1>
Which country is the magazine Woman's Viewpoint published in?
</SUB-QUESTION-1>
<CORRESPONDING-DOC-1>
RELATED-DOC-2
</CORRESPONDING-DOC-1>
<SUB-ANSWER-1>
The magazine 'Woman's Viewpoint' was published in Texas, a state located in the United States.
</SUB-ANSWER-1>
<SUB-QUESTION-2>
Which country is the magazine Pick Me Up published in?
</SUB-QUESTION-2>
<CORRESPONDING-DOC-2>
RELATED-DOC-1
</CORRESPONDING-DOC-2>
<SUB-ANSWER-2>
The magazine Pick Me Up! is published in the United Kingdom.
</SUB-ANSWER-2>
<FINAL-ANSWER>
No, they are not both British publications. "Pick Me Up!" is indeed a British magazine published in the United Kingdom. However, "Woman's Viewpoint" was published in Texas, which is in the United States, so it is an American publication.
</FINAL-ANSWER>

Here is an input you need to process:
<QUESTION> {question} </QUESTION>
<ANSWER> {answer} </ANSWER>
{related_doc}

Please format your output as shown above and refrain from including any additional content.

Figure 13: Prompt for constructing data in Multi-hop QA setting, using Qwen72B.

H G → R Task: Entity Linking

This section presents the details of the **G → R** task, including the detailed construction of the data, training details, inference details, and additional experimental results.

H.1 Training Details

A comprehensive example of training Entity Linking is illustrated in Figure 6 (a).

Data Reconstruction. Here, we outline the adaptations we implemented in the training dataset for Entity Linking. Initially, we demonstrate how LLMs can be utilized to perform the Mention Detection task in a generative fashion. As shown in the following diagram, the input comprises specific extraction instructions alongside the sentence targeted for extraction, and the output is a sentence annotated accordingly. All outputs contribute to the calculation of the generative loss.

Input: <Instruction>. Steve Jobs founded Apple Inc.
Output: <MENTION> Steve Jobs</MENTION> founded <MENTION>Apple Inc</MENTION>.

For this case, we append two special tokens, [RQ] and [CON], subsequent to each </MENTION> token. The [RQ] token is intended to prompt the model to extract semantic information from the preceding mention, whereas the [CON] token aids in the LLMs' generation of subsequent content. It should be noted that only the [RQ] token is considered in the computation of representative loss, while the [CON] token remains involved in the generative loss calculation.

Input: <Instruction>. Steve Jobs founded Apple Inc.
Output: <MENTION>Steve Jobs</MENTION> [RQ] [CON] founded <MENTION>Apple Inc</MENTION> [RQ] [CON] .

The computation of representative loss necessitates the use of both positive and negative examples. Here is a set of positive and negative examples about Steve Jobs:

Positive:
Steven Paul Jobs (February 24, 1955 – October 5, 2011) was an American businessman, inventor, and investor best known for co-founding the technology giant Apple Inc. [RD]
Negative:
Steve Jobs is a 2015 biographical drama film directed by Danny Boyle and written by Aaron Sorkin. A British-American co-production, it was adapted from the 2011 biography by Walter Isaacson and interviews conducted by Sorkin. [RD]

Figure 14: A case for data reconstruction of positive document and negative document about Entity Linking. Add [RD] token at the end of each document.

In these examples, the [RQ] token directs the LLMs to extract semantic information from the current document. Only the [RQ] token participates in the representative loss computation, with the other elements excluded from this process.

Implementation Details. We conducted our training using the AIDA dataset combined with 1% of Wikipedia data, yielding a total of approximately 68k samples. The training is performed on eight A800 machines, leveraging the DeepSpeed ZeRO-3 strategy to optimize memory utilization. We configured the maximum sequence length at 1300 and established a batch size of 5 per GPU, which resulted in an overall global batch size of

40. Each sentence in the dataset contained multiple [RQ] tokens, from which we randomly selected one token per sentence for optimization, paired with one positive and four negative samples. The negative samples were shared across the batch. The training regimen included 5k steps with a learning rate of 5e-6.

H.2 Inference Details

Algorithm H.3 presents the pseudocode for Entity Linking inference. Figure 7(a) provides a schematic representation of the Entity Linking inference process.

Candidate Construction. Our consolidated entity library comprises five distinct repositories:

- Utilizing ReFinED (Ayoola et al., 2022), we generate *Candidate Repository 1* from the training datasets based on annotated mentions.
- We utilize *Candidate Repository 2* for the test datasets, sourced from ChatEL (Ding et al., 2024b), which is annotated by REL (van Hulst et al., 2020) and BLINK (Wu et al., 2020).
- *Candidate Repository 3* is included, provided by (Hoffart et al., 2011).
- Through the Tool from (Lai et al., 2022), we extract the top 10 challenging candidates from both the AIDA training and test datasets within the Wikidata repository, resulting in *Candidate Repository 4*.
- *Candidate Repository 5* is created by applying ReFinED to mentions extracted by our method.

H.3 Evaluation Details

Baselines. For EL, we employ REL 2019 (van Hulst et al., 2020), Neural EL (Kolitsas et al., 2018), GENRE (Cao et al., 2021b) and ReFinED (Ayoola et al., 2022) as our baselines. For MD, we expand EL baselines to include LLMs such as Qwen-7B-chat, Qwen-14B-chat (Bai et al., 2023), Llama2-7B-chat, and Llama3-7B-chat. For ED, we expand EL baselines to include ChatEL (Ding et al., 2024b) and EntGPT (Ding et al., 2024a). ChatEL transforms the ED task into a question-answering task, utilizing prompts to facilitate GPT-4's execution of the task. Conversely, EntGPT involves fine-tuning GPT-3.5 (OpenAI, 2022) using specially constructed datasets.

Evaluation Setup, Datasets, and Metric. For EL task, we utilize the ELEVANT (Bast et al., 2022) tool for evaluation across seven datasets: the in-domain AIDA-CoNLL (AIDA) and six out-of-domain datasets, MSNBC (Cucerzan, 2007)(MSN), KORE50 (Hoffart et al., 2012)(K50), N3-Reuters-128 (Röder et al., 2014)(REU), SpotLight (SPOT), OKE Challenge 2015 (Nuzzolese et al., 2015)(O15), and OKE Challenge 2016 (Nuzzolese et al., 2016) (O16), using the Micro F1 metric to evaluate in-KB entities. The same datasets and metrics are applied to the MD task. For ED task, following the ChatEL (Ding et al., 2024b), we extend our evaluation to include the RSS (Röder et al., 2014) and ACE04 datasets, maintaining the use of the Micro F1 metric. In assessing candidate entities for each mention in EL task, our methodology leverages a proprietary entity repository of 1.25M entities derived from Wikipedia and Wikidata, which includes numerous indistinguishable entities. For baseline comparisons, the ELEVANT (Bast et al., 2022) tool is also employed.

H.4 Experiments

H.4.1 Efficiency Settings

We conduct tests on a single 40GB A100 card, with an `Intel(R) Xeon(R) Platinum 8352V CPU @ 2.10GHz`. For the configuration of the pipeline, it involves two LLMs: one for extracting mentions from sentences and another for linking based on the extracted mentions. The extraction process is driven by an instruction that includes the target sentence and a specific extraction instruction, resulting in a sentence annotated with annotation. The linking process is structured as a question-answering (QA) task, utilizing a prompt composed of the sentence, a linking instruction, a list of four candidates. The output for linking is set to a single token. In the OneGen configuration, ours allow for direct retrieval during the generation process. Consequently, the number of output tokens surpasses those from the pipeline's extraction output by an increment of $2n$ tokens, where n represents the number of mentions in the sentence. Specifically, the sentence intended for extraction is limited to 1000 tokens, while the instructions for both extraction and linking are capped at 15 tokens each, and each candidate description is confined to 30 tokens.

H.4.2 Ablation

Recall Strategy for candidate sets. In the EL task, once a mention is extracted, it is necessary to select

Algorithm 2 Entity Linking Inference

```
Input:
    LLM trained with OneGen, denoted as $\hat{f}(\cdot)$
    LLM without the LM-Head, denoted as $f(\cdot)$
    Pre-cached document vector library $Emb_{doc}$
    Instruction with text to be extracted $x$
    Cosine similarity computation function CosineSimilarity()
    Function to sort and return indices Top1()
Output:
    Text with marked mentions $History$
    List of entities id corresponding to the mentions in the text $EntityList$

$History \leftarrow x$
$EntityList \leftarrow []$
$NextToken \leftarrow \hat{f}(History)$
while $NextToken \notin$ Terminator do
    $History \leftarrow History \cup \{NextToken\}$
    if $role(NextToken) =$ RET then                                  ▷ Retrieval on demand
        $scores \leftarrow$ CosineSimilarity($f(History), Emb_{doc}$)
        $EntityList \leftarrow EntityList \cup \{$Top1($scores$)$\}$
        $History \leftarrow History \cup \{$[CON]$\}$
    end if
    $NextToken \leftarrow \hat{f}(History)$                              ▷ Generation
end while
return $History, EntityList$
```

Method	Recall Strategy	AVG.
ReFinED	ReFinED	60.8
Llama2$_{7B}$+OneGen	ReFinED	<u>61.8</u>
Llama2$_{7B}$+OneGen	Ours	**64.0**

Table 12: Ablation study results of recall strategies for entity linking, reporting average F1 scores across seven datasets.

the corret entity corresponding to the mention from a candidate entity repository. Our previous method build a challenging repository that allow OneGen to choose from 1.25M entities without additional recall strategy, achieving 100% recall on the test set. For ablation, we use the ReFinED recall strategy, which maps mentions to 30 potential entities using a rule-based dictionary, with a recall rate of 96% on the test set. Our evaluations across seven EL datasets (reported in Table 12) demonstrate that OneGen consistently outperforms ReFinED, indicating that improvements in the retrieval process significantly enhance EL performance.

Method	**Training Data**	**Cand. Size**	K50	OKE15	OKE16	REU	RSS	ACE04	MSN	WIKI	AQU	**AVG.**
					Baselines							
REL	-	<30	61.8	70.5	74.9	66.2	68.0	89.7	93.0	78.3	88.1	76.7
Neural EL	-		76.7	78.3	67.7	72.0	88.0	92.0	74.0	88.0	77.1	
GENRE	WIKI 6M+AIDA		54.2	64.0	70.8	69.7	70.8	84.8	78.0	82.3	84.9	73.3
ReFinED	WIKI 6M+AIDA		56.7	78.1	79.4	68.0	70.8	86.4	89.1	84.1	86.1	77.6
					LLMs Baselines							
ChatEL (GPT-4)	Prompt	<30	**78.7**	75.8	75.2	78.9	82.2	89.3	88.1	79.1	76.7	80.4
EntGPT-P (GPT-3.5)	AIDA		71.6	76.7	77.0	78.5	80.8	91.8	86.7	80.8	79.1	80.3
EntGPT-I (GPT-3.5)	AIDA		75.3	82.5	81.9	80.8	82.5	**93.7**	92.2	79.1	**90.6**	84.3
					Our Method							
Llama2$_{7B}$(+OneGen)	WIKI 60K+AIDA	1.25M	77.0	**87.5**	**87.5**	**85.2**	**85.3**	92.2	**92.5**	**85.5**	86.0	**86.5**

Table 13: Entity Disambiguation InKB micro F1 scores on in-domain and out-of-domain test sets. The best value in bold and second best is underlined. The results of the baselines come from ChatEL (Ding et al., 2024b) and EntGPT (Ding et al., 2024a). The dataset used here differs slightly from the dataset used for Entity Linking. For details, refer to ChatEL.

Method	**Training Data**	**In-Domain**	**Out-of-domain**						**AVG.**
		AIDA	OKE15	OKE16	REU	MSN	SPOT	KORE50	
				Baselines					
REL2014	-	90.3	67.2	58.8	61.8	82.6	27.7	95.2	69.1
REL2019	-	90.5	67.5	58.8	62.3	82.8	27.7	95.2	69.3
Neural EL	AIDA	95.8	68.6	60.3	71.4	79.3	23.4	82.1	68.7
GENRE	Wiki 6M + AIDA	85.5	54.4	47.5	45.6	68.5	26.9	83.3	58.8
ReFinED	Wiki 6M + AIDA	**95.9**	**70.5**	61.9	**76.7**	**84.2**	24.0	**95.9**	**72.7**
				LLMs Baselines using SFT					
Qwen$_{7B}$	Wiki 60K + AIDA	85.1	65.7	65.1	69.4	79.5	41.1	94.0	71.4
Qwen$_{14B}$		91.2	66.2	65.8	71.1	78.3	**41.8**	93.4	72.6
Llama3$_{7B}$		88.5	67.3	**65.9**	69.4	75.6	37.1	92.9	71.0
Llama2$_{7B}$		85.6	68.6	63.9	74.9	80.6	31.4	92.9	71.1
				Ours					
Llama2$_{7B}$(+OneGen)	Wiki 60K + AIDA	88.6	66.6	64.5	74.7	80.5	33.5	92.4	71.5

Table 14: Mention Detection micro F1 scores on in-domain and out-of-domain test sets. The best value in bold and second best is underlined.