DiffuGR: Generative Document Retrieval with Diffusion Language Models

Xinpeng Zhao
zhaoxp1001@gmail.com
Shandong University
Shandong, China

Zhaochun Ren
z.ren@liacs.leidenuniv.nl
Leiden University
Leiden, Netherlands

Yukun Zhao
yunkunzhao.sdu@gmail.com
Baidu Inc.
Beijing, China

Zhenyang Li
zhenyounglee@gmail.com
Baidu Inc.
Beijing, China

Mengqi Zhang
mengqi.zhang@sdu.edu.cn
Shandong University
Shandong, China

Jun Feng
junfeng0288@gmail.com
Chinese Academy of Sciences
Beijing, China

Ran Chen
chenran@stu.pku.edu.cn
Peking University
Beijing, China

Ying Zhou
yingzhou@sdu.edu.cn
Shandong University
Shandong, China

Zhumin Chen
chenzhumin@sdu.edu.cn
Shandong University
Shandong, China

Shuaiqiang Wang
shqiang.wang@gmail.com
Baidu Inc.
Beijing, China

Dawei Yin
yindawei@acm.org
Baidu Inc.
Beijing, China

Xin Xin*
xinxin@sdu.edu.cn
Shandong University
Shandong, China

Abstract

Generative retrieval (GR) re-frames document retrieval as a sequence-based document identifier (DocID) generation task, memorizing documents with model parameters and enabling end-to-end retrieval without explicit indexing. Existing GR methods are based on auto-regressive generative models, i.e., the token generation is performed from left to right. However, such auto-regressive methods suffer from: (1) mismatch between DocID generation and natural language generation, e.g., an incorrect DocID token generated in early left steps would lead to totally erroneous retrieval; and (2) failure to balance the trade-off between retrieval efficiency and accuracy dynamically, which is crucial for practical applications.

To address these limitations, we propose generative document retrieval with diffusion language models, dubbed DiffuGR. It models DocID generation as a discrete diffusion process: during training, DocIDs are corrupted through a stochastic masking process, and a diffusion language model is learned to recover them under a retrieval-aware objective. For inference, DiffuGR attempts to generate DocID tokens in parallel and refines them through a controllable number of denoising steps. In contrast to conventional left-to-right auto-regressive decoding, DiffuGR provides a novel mechanism to first generate more confident DocID tokens and refine the generation through diffusion-based denoising. Moreover, DiffuGR also offers explicit runtime control over the quality–latency tradeoff. Extensive experiments on benchmark retrieval datasets show that DiffuGR is competitive with strong auto-regressive generative retrievers, while offering flexible speed and accuracy tradeoffs through variable denoising budgets. Overall, our results indicate that non-autoregressive diffusion models are a practical and effective alternative for generative document retrieval.[1]

*Corresponding author.

Conference'17, Washington, DC, USA

ACM ISBN 978-1-4503-XXXX-X/2026/04
https://doi.org/XXXXXXX.XXXXXXX

CCS Concepts

• **Information systems** → **Information retrieval**; **Retrieval models and ranking**; **Novelty in information retrieval**.

Keywords

Generative Retrieval, Diffusion Models, Document Retrieval

ACM Reference Format:
Xinpeng Zhao, Zhaochun Ren, Yukun Zhao, Zhenyang Li, Mengqi Zhang, Jun Feng, Ran Chen, Ying Zhou, Zhumin Chen, Shuaiqiang Wang, Dawei Yin, and Xin Xin. 2026. DiffuGR: Generative Document Retrieval with Diffusion Language Models. In . ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

Relevance Statement. This paper proposes a new diffusion-based generative document retrieval method DiffuGR. It aligns directly with the Web track topic of **Search and retrieval-augmented AI** and can be deployed in real-world Web systems, such as search engine, to obtain better search results.

[1]This paper is under review.

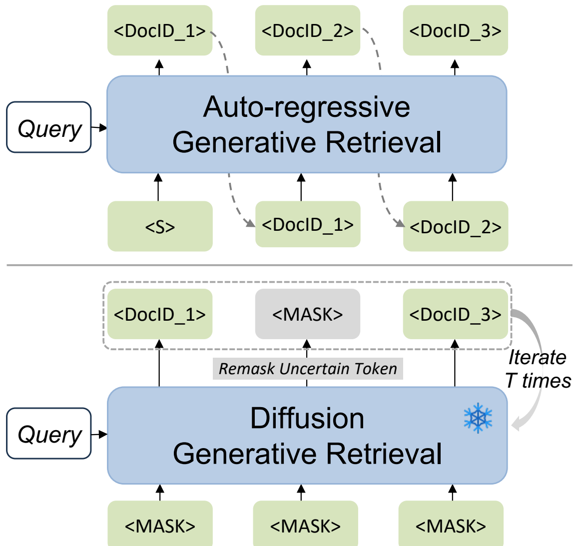

Figure 1: In contrast to auto-regressive generative retrieval (above), which generates the DocID from left to right, DiffuGR (below) efficiently generates multiple tokens in parallel and then re-masks uncertain tokens. The generation is progressively refined over multiple denoising steps.

1 Introduction

Generative Retrieval. Information retrieval (IR) is a fundamental component in modern artificial intelligence systems, powering search engines, recommendation systems, and large-scale knowledge access. In recent years, generative retrieval (GR) has emerged as a promising paradigm that replaces traditional index-based retrieval pipelines with an end-to-end generative model capable of directly producing document identifiers (DocIDs) in response to a query. GR first assigns each document with a unique sequence of tokens, aka DocID, then a generative sequence-to-sequence model is utilized to mapping the input sequence of a query to the output sequence of DocID tokens. By integrating retrieval tightly with generation and memorizing the document with model parameters, GR offers new potential to unify retrieval and reasoning, reduce system complexity, and unlock more flexible knowledge access.

Auto-regressive Generation for GR. Most existing GR methods are built upon auto-regressive language models, which sequentially generate discrete identifier tokens in a left-to-right manner, as shown in the above part of Figure 1. Tay et al. [42] first proposed a differentiable search index (DSI) for GR, which utilizes a T5 language model as the generator. Plenty of research has been conducted following this research line, including constructing better DocIDs [31, 40, 47], using larger model parameters [4, 36], and exploring new training and decoding strategies [40, 45].

Existing Limitations. Although these approaches have demonstrated effectiveness in achieving competitive performance compared to dense retrieval baselines, following limitations still exist:

(1) *Mismatch between DocID generation and natural language generation.* Different with natural language generation where the sequential order of words contains specific semantic information, tokens of a DocID may not have so strong left-to-right dependencies. It indicates that the auto-regressive left-to-right generation of DocID tokens could limit the retrieval performance. For example, if an incorrect DocID token is generated in early left steps, the following decoding generation steps would become meaningless since the correct DocID would never be generated.

(2) *Failure to dynamically control the trade-off between efficiency and accuracy.* Existing auto-regressive based GR methods can only generate DocID tokens one by one during inference with squared increasing computational cost. However, the capability to dynamically control the trade-off between inference efficiency and retrieval accuracy is of vital importance for real-world scenarios.

The Proposed Method. To address these limitations, we proposed generative document retrieval with diffusion language models, dubbed DiffuGR, which models DocID generation as a discrete diffusion process. Specifically, for each document, DiffuGR first assigns a specific sequence of tokens as the DocID for this document. DiffuGR involves two kinds of DocIDs, including learnable DocIDs and linguistic DocIDs. Then, for model training, DocIDs are corrupted through a stochastic masking process, and a diffusion language model is learned to recover the masked tokens under a retrieval-aware objective function. At the inference stage, given an input query, DiffuGR generates DocID tokens in parallel and refines them through a controllable number of denoising steps, as shown in the bottom part of Figure 1.

DiffuGR offers several appealing properties for GR compared to auto-regressive models. Firstly, its diffusion-based sampling enables parallel generation, where candidate DocID tokens are iteratively denoised rather than generated from left to right, allowing the model to revise and correct earlier errors instead of being constrained by erroneous token commitments. As a result, DiffuGR would first confirm more confident DocID tokens and gradually refine the generation across multiple denoising steps. Secondly, DiffuGR enables dynamic control of the trade-off between efficiency and accuracy through adjusting the number of denoising steps. For example, less denoising steps indicate faster inference speed while more denoising steps lead to higher retrieval accuracy. Besides, the stochastic nature of denoising allows DiffuGR to produce multiple plausible candidate DocIDs, thereby enhancing both the diversity and robustness of the retrieval results. Through these advantages, DiffuGR not only mitigates the generation mismatch but also opens up new possibilities for flexible quality–latency control.

Main Findings. In this paper, we conduct systematic study of DiffuGR. Our key findings are:

(1) *Comparable performance.* DiffuGR has achieved competitive retrieval performance on two widely used datasets, demonstrating the effectiveness of the proposed method.

(2) *Flexible runtime control.* DiffuGR offers flexible runtime control over the quality-latency trade-off, which enables retrieval systems to perform dynamically scheduling based on load conditions.

(3) *Existing weakness.* Although experimental results have demonstrated the effectiveness of DiffuGR, the parallel generation indicates that DiffuGR does not support beam search, leading to trivial improvement on Recall@10 and Recall@100. It's necessary to develop new methods for top-k generation under diffusion scenarios.

Main Contributions. Our contributions can be summarized as:

- We introduce diffusion language models as a new paradigm for GR, encouraging non-autoregressive DocID generation. To the best of our knowledge, this is the first work to utilize diffusion language models for end-to-end document retrieval.
- We conduct extensive empirical comparisons between diffusion-based and auto-regressive GR, highlighting their respective strengths and weaknesses.
- We demonstrate that DiffuGR not only matches the performance of auto-regressive GR methods, but also offers unique advantages of flexible quality-latency control.

To summarize, our work positions diffusion language models as a feasible and potentially superior alternative to auto-regressive models for GR. We hope this study opens up a new line of research at the intersection of generative modeling and information retrieval.

2 Related Work

In this section, we give a literature review regarding generative retrieval and diffusion models for text generation.

2.1 Generative Retrieval

Generative retrieval (GR) reformulates document retrieval as a generation task, in which a generative model (usually a seq2seq language model) directly generates the identifier of the target document. It provides an end-to-end solution for document retrieval and allows better utilization of large-scale generative language models. Document identifiers (DocIDs) are important for GR, through which retrieval systems can uniquely index the corresponding documents and enable end-to-end training and inference. Existing GR methods usually use learnable DocIDs or linguistic DocIDs.

Learnable DocIDs are usually constructed upon dense document representations, either through hierarchical k-means clustering [42, 44], product quantization [32, 60], residual vector quantization [38, 53, 55], or progressively learning [40, 47]. This process targets on compressing the semantic information contained in dense document representations into discrete tokens. Although promising results have been achieved, learnable DocIDs require expansion of the vocabulary, e.g., introducing additional trainable codebooks.

Linguistic DocIDs use strings, such as title, URL, or keywords, to represent documents. They naturally carry semantics related to associated documents, and the construction cost is relatively low, without the need for additional manual supervision. Plenty of work has been conducted based on linguistic DocIDs, yielding excellent results[3, 7, 9, 10, 12, 25, 41, 56, 57, 60].

Despite the success of GR, most existing methods generate DocID tokens in an auto-regressive left-to-right manner. Although Qiao et al. [37] proposed DiffusionRet which employs the diffusion model to generate a pseudo document for a query, the generation of DocIDs is still auto-regressive. However, such auto-regressive left-to-right generation of DocID tokens could limit the retrieval performance. Specifically, if an incorrect DocID token is generated in early left steps, the correct DocID would never be generated. Based on the above considerations, this work explores end-to-end generation of DocIDs through non-autoregressive diffusion language models [34, 51]. Its core advantage lies in the fact that the diffusion model does not restrict the left-to-right generation order; instead, the model is encouraged to automatically select the most appropriate generation order based on its own conditions and confidence.

2.2 Diffusion Models for Text Generation

Diffusion models are initially introduced in continuous domains such as image generation [19, 39], and their success has inspired extensions to natural language processing by modeling text in continuous embedding spaces [15, 16, 26]. These continuous diffusion language models demonstrate the feasibility of applying iterative denoising for text generation, for both pre-training and fine-tuning on transformer architectures [29]. To better align with the discrete nature of text, discrete diffusion language models were later developed [2, 5, 20], which progressively corrupt token sequences and then reconstruct the original text during the reverse process. Further benefits have been observed through initializing from pre-trained masked language models such as BERT [18, 52]. Despite generation quality, recent work highlights the strengths of diffusion models in hybrid block-wise generation that balance sequential coherence with parallelism [1, 48–50].

Scaling diffusion models to large language models (LLMs) has further extended their applicability. Gulrajani and Hashimoto [17] analyzed scaling laws for continuous diffusion. Lou et al. [30] showed that discrete masked diffusion models have achieved perplexities competitive with GPT-2. Building on this, large-scale efforts such as DiffuGPT and DiffuLLaMA adopt pretrained auto-regressive LLMs into diffusion-based frameworks [14], while LLaDA [34] demonstrated that diffusion models trained from scratch at 8B parameters can rival strong auto-regressive LLMs like LLaMA3-8B. Recent commercial systems such as Mercury Coder [21] further validate the practicality of diffusion models for text and code generation.

In summary, these advances establish diffusion models as a viable alternative for natural language generation. While in this work, we explore the new potential to use diffusion models for end-to-end document retrieval, i.e., the generation of DocIDs for a query.

3 Method

In this section, we present details of the proposed DiffuGR. We first formulate the task of generative document retrieval in Section 3.1. The construction of DocIDs is illustrated in Section 3.2, then the training procedure of DiffuGR is described in Section 3.3. Finally, model inference is detailed in Section 3.4. Figure 2 shows an overall view of the proposed DiffuGR.

3.1 Task Formulation

This paper focuses on the task of generative retrieval, where the goal is to directly generate a document identifier (Docid) given a user query. Formally, let $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$ denote the document collection, and $\mathcal{Q} = \{q_1, q_2, \ldots\}$ denote user queries. Each document d is associated with a sequence of discrete tokens (DocID) $\mathbf{z} = \{z_1, z_2, \ldots, z_l\}$, where l is the length of DocIDs. Given a query q, the model is required to generate the corresponding $\mathbf{z}$ that identifies the relevant document.

3.2 Construction of DocIDs

A key factor in GR is the design of DocIDs. Unlike dense retrieval methods, where continuous vectors implicitly encode document

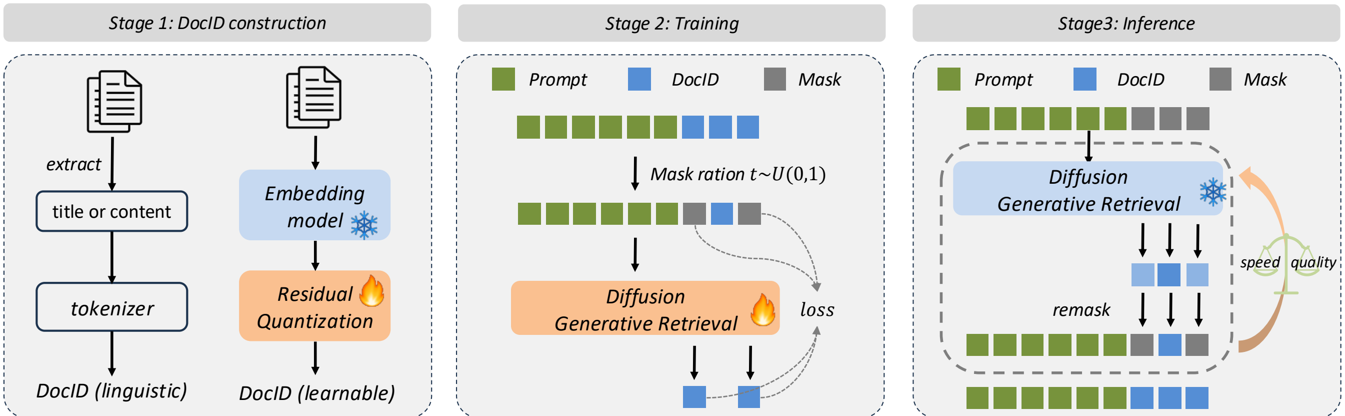

Figure 2: An overview of DiffuGR, consisting DocID construction, model training, and inference. DiffuGR involves two kinds of DocIDs, i.e., linguistic DocIDs and learnable DocIDs. During training, the model is optimized to recover randomly masked DocID tokens. For inference, DiffuGR generates DocID tokens in parallel and refines them across multiple denoising steps.

semantics, GR requires DocIDs to be discrete, compact, and decodable. To systematically study the effect of DocIDs for DiffuGR, we explore both learnable DocIDs and n-gram based linguistic DocIDs.

3.2.1 Learnable DocIDs. In this setting, DocIDs are learned through residual quantization over dense document embeddings. Let $\mathbf{d}$ denotes the embedding of document d. The initial residual vector is defined as $\mathbf{r}_0 = \mathbf{d}$. The quantization is conducted recursively. At i-th step, we find the nearest neighbor within the i-th codebook $\mathbf{E}_i = \{\mathbf{e}_i^k \in \mathbb{R}^d \mid k = 1, \ldots, K_i\}$, where K_i is the size of the i-th codebook, selecting the closest code embedding $\mathbf{e}_i^{z_i}$ to the current residual vector:

$$z_i = \underset{k \in K_i}{\operatorname{argmin}} ||\mathbf{r}_{i-1} - \mathbf{e}_i^k||^2. \tag{1}$$

The residual vector is updated as:

$$\mathbf{r}_i = \mathbf{r}_{i-1} - \mathbf{e}_i^{z_i}, \tag{2}$$

The original embedding $\mathbf{d}$ is approximated by adding the code embeddings. For training of the i-th codebook, we adopt the following loss:

$$\mathcal{L}_{\text{rq}} = \sum_{\mathcal{D}} \sum_{i=1}^{l} ||\mathbf{r}_{i-1} - \mathbf{e}_i^{z_i}||^2. \tag{3}$$

3.2.2 Linguistic DocIDs. In addition to learnable identifiers, we investigate n-gram based DocIDs, which are directly constructed from the text of each document. These DocIDs are lightweight, interpretable, and vocabulary-aligned, making them well-suited for language model generation. We consider two representative variants, including:

(1) *Title-based DocIDs.* The title of each document is directly adopted as the DocID for this document. Titles are typically concise human-written summaries of the main content, and hence naturally capture the document's intent. However, not all documents have well-formed or unique titles. To address such issue, DiffuGR incorporates the second kind of DocIDs.

(2) *Leading-token DocIDs.* The first n tokens of the document content are regarded as the DocID for this document. This alternative exploits the observation that the opening part of a document often introduces its main topic. It should be noted that here we treat such DocIDs as a supplementary method, which is only used to handle samples without document titles.

With the two simple methods mentioned above, we can easily construct DocIDs for DiffuGR. Note that the focus of this paper lies in non-autoregressive generation of DocIDs other than the construction. Therefore, using such simple DocID construction methods allows us to better validate the effectiveness of diffusion-based generation. As for how to construct more suitable DocIDs for diffusion models, we will explore this problem in future works.

3.3 Diffusion for Generative Retrieval

To generate DocIDs for a given query, DiffuGR employs masked diffusion language models, which differ from standard auto-regressive models. Specifically, instead of predicting the next token sequentially, the masked diffusion language model learns to iteratively denoise partially masked sequences until a complete identifier sequence is obtained. Masked diffusion language models introduce a forward process that gradually adds noise to the sequence and learn a corresponding reverse process to generate DocIDs.

3.3.1 Forward process. Let V and L denote the vocabulary size of a masked diffusion language model and the length of input sequence, respectively. Given a sequence $\boldsymbol{x}_0 \in \{0, 1, \ldots, V-1\}^L$ and a noise level $t \in [0, 1]$, the forward process randomly and independently masks the tokens in the sequence, formulated as follows:

$$q_{t|0}(\boldsymbol{x}_t|\boldsymbol{x}_0) = \prod_{i=0}^{L-1} q_{t|0}(\boldsymbol{x}_t^i|\boldsymbol{x}_0^i) \tag{4}$$

$$q_{t|0}(\boldsymbol{x}_t^i|\boldsymbol{x}_0^i) = \begin{cases} 1-t, & \boldsymbol{x}_t^i = \boldsymbol{x}_0^i, \\ t, & \boldsymbol{x}_t^i = m, \end{cases} \tag{5}$$

where $\boldsymbol{x}^i$ denotes the i-th element of $\boldsymbol{x}$, m denotes the mask token [13], $\boldsymbol{x}_t$ denotes the noisy data at time t and $q_0(\cdot)$ is the data distribution $p_{\text{data}}(\cdot)$.

3.3.2 Reverse process. The reverse process iteratively recovers tokens for masked positions, starting from a mask sequence $\boldsymbol{x}_1$. Let

$0 \le s < t \le 1$, the reverse process is characterized by

$$q_{s|t}(\boldsymbol{x}_s|\boldsymbol{x}_t) = \prod_{i=0}^{L-1} q_{s|t}(\boldsymbol{x}_s^i|\boldsymbol{x}_t) \tag{6}$$

$$q_{s|t}(\boldsymbol{x}_s^i|\boldsymbol{x}_t) = \begin{cases} 1, & \boldsymbol{x}_t^i \neq m, \boldsymbol{x}_s^i = \boldsymbol{x}_t^i, \\ \frac{s}{t}, & \boldsymbol{x}_t^i = m, \boldsymbol{x}_s^i = m, \\ \frac{t-s}{t} q_{0|t}(\boldsymbol{x}_s^i|\boldsymbol{x}_t), & \boldsymbol{x}_t^i = m, \boldsymbol{x}_s^i \neq m, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Here $q_{0|t}(\cdot|\cdot)$ is the data prediction model to be learned. Notably, Ou et al. [35] revealed an intrinsic property that $q_{0|t}(\cdot|\cdot)$ can be represented by conditional distributions on clean data $p_{\text{data}}(\cdot|\cdot)$ independently from t. Formally,

$$q_{0|t}(\boldsymbol{x}_0^i|\boldsymbol{x}_t) = p_{\text{data}}(\boldsymbol{x}_0^i|\boldsymbol{x}_t^{\text{UM}}), \tag{8}$$

where $\boldsymbol{x}_t^{\text{UM}}$ collects all unmasked tokens in noisy data $\boldsymbol{x}_t$ and $p_{\text{data}}(\cdot|\cdot)$ is irrelevant to t.[2]

3.3.3 Training Objective. The model is trained to reconstruct the original sequence by predicting the masked tokens. A distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}_0^i|\boldsymbol{x}_t)$ parameterized by $\boldsymbol{\theta}$ is employed to approximate $p_{\text{data}}(\boldsymbol{x}_0^i|\boldsymbol{x}_t^{\text{UM}})$, optimizing the following upper bound on negative log-likelihood:

$$-\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0) \le \int_0^1 \frac{1}{t} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} \left[\sum_{\{i|\boldsymbol{x}_t^i = m\}} -\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0^i|\boldsymbol{x}_t) \right] dt \triangleq \mathcal{L}. \tag{9}$$

To put it simply, given a sequence $\boldsymbol{x}_0$, we randomly mask a subset of tokens according to a corruption process $q_{t|0}(\boldsymbol{x}_t|\boldsymbol{x}_0)$, where $\boldsymbol{x}_t$ denotes the corrupted sequence at time t. This objective resembles denoising autoencoding, but the iterative refinement aligns naturally with the diffusion process, where the model progressively recovers signal from noise. As for generative retrieval task, the training data contains query-DocID pairs $(q, \mathbf{z})$ and document-DocID pairs $(d, \mathbf{z})$. We only randomly mask the DocID sequences $\mathbf{z}$ for training the masked diffusion language model. During training, both categories of DocIDs are used to construct supervision pairs $(q, \mathbf{z})$. Queries are paired with their corresponding document identifiers, and the model learns to generate $\mathbf{z}$ from q.

3.4 Model Inference

At inference time, the query is concatenated with fully masked DocID placeholders, and the model attempts to generate DocID tokens in parallel and refine the generation through diffusion-based denoising. The total number of denoising iteration is a hyperparameter, which naturally provides diffusion language models with a trade-off between efficiency and quality. Following Nie et al. [34], Ye et al. [51], we employ uniformly distributed t by default. Due to the inherent properties of the masked diffusion language model, the generation length l is also a predefined hyperparameter. In our work, we set l to the maximum length of the DocIDs.

[2]For example, if $\boldsymbol{x}_t = [1, 3, m, 2]$, then $\boldsymbol{x}_t^{\text{UM}} = [1, 3, \cdot, 2]$ and $p_{\text{data}}(\cdot|[1, 3, \cdot, 2])$ is irrelevant to t.

3.4.1 Diffusion Denoising Strategies. At the first step, we feed both q and $\mathbf{z}^1$, where the superscript of $\mathbf{z}^1$ means that the DocID tokens are all masked. Then, for steps from time $t \in (0, 1]$ to $s \in [0, t)$, we feed both q and $\mathbf{z}^s$ into the masked diffusion language model. Here, $\mathbf{z}^s$ denotes the DocID tokens in timestamp s, where some tokens remain masked. The model then predicts all currently masked DocID tokens simultaneously. Subsequently, DiffuGR re-masks a fraction of the newly predicted tokens, with an expected proportion of s/t. This ensures that the transition in the reverse process is consistent with the forward process, enabling accurate sampling.

When it comes to specific implementation of re-masking, a variety of different strategies can be used. In this work, DiffuGR involves four denoising scheduling methods that determine which token to be re-masked and regenerated during the reverse diffusion process:

(1) *Random.* The random decoding strategy follows a purely random generation order [2], which serves as the default baseline for diffusion-based language models. This strategy introduces maximal stochasticity into the denoising trajectory but may lead to suboptimal recovery in structured tasks such as GR.

(2) *Maskgit plus.* This method adopts the confidence-based token scheduling from [8], where tokens with the lowest predicted confidence are re-masked. This deterministic progression encourages stable convergence and reduces accumulation of early-stage errors.

(3) *Top-k margin.* This method extends the confidence-based scheduling by ranking tokens according to margin confidence (i.e., the difference between top-1 and top-2 probabilities) [23]. This method emphasizes the relative certainty of token predictions, prioritizing positions with more decisive model beliefs for generation.

(4) *Entropy* This method computes the entropy of the token probability distribution and re-masks tokens with the highest entropy, effectively denoising regions where the model exhibits higher confidence firstly. Compared to the margin-based strategy, entropy-based scheduling provides a smoother and information-theoretic measure for generation confidence.

Compared to auto-regressive decoding, such above paradigm offers more flexibility in parallel token updates, and is well-suited for generating structured DocIDs with multiple discrete components. At inference time, given a query, the model outputs a DocID, which is then mapped to the corresponding document in $\mathcal{D}$. In Section 5.4.1, experiments were conducted to investigate the influence of different denoising strategies.

3.4.2 Pseudo Beam Search for DiffuGR. While DiffuGR offers efficient parallel token generation, it inherently lacks the sequential tree structure that enables candidate exploration in beam search. Consequently,DiffuGR cannot directly leverage beam expansion to produce multiple candidate DocIDs. To address this limitation, we propose two complementary strategies that approximate the effect of beam search within the diffusion-based framework while maintaining its parallel decoding nature.

The first strategy leverages query augmentation, where a single query is reformulated into multiple semantically equivalent but lexically diverse variants through paraphrasing models. These augmented queries are jointly fed into the diffusion generator as independent inputs. Each variant undergoes the same denoising trajectory, yielding a set of candidate DocIDs that capture multiple plausible retrieval hypotheses. This procedure effectively mimics

Table 1: Statistics of datasets.

Dataset	# Docs	# Test queries	# Train pairs
NQ320K	109,739	7,830	307,373
MS MARCO	323,569	5,187	366,235

beam search exploration by increasing input diversity rather than branching during generation. In this paper, DeepSeek-V3-0324 is used to conduct the query augmentation.

The second strategy exploits intermediate denoising states within the diffusion process itself. Given a total of T denoising steps, the model produces T intermediate DocID candidates corresponding to different noise levels. By retaining and evaluating these intermediate outputs, DiffuGR can explore multiple decoding hypotheses at negligible additional cost. However, these intermediate DocIDs may contain duplicates or invalid tokens due to stochastic denoising noise, necessitating a lightweight post-filtering mechanism.

The above techniques enable DiffuGR to approximate candidate diversity of beam search without sacrificing its parallel generation property, thus improving retrieval robustness and recall metrics.

4 Experimental Setup

In this section, we describe the detail of experimental settings.

4.1 Datasets

We conduct experiments on two well-known document retrieval datasets: NQ [24] and MS MARCO [6]. Statistics of these datasets are summarized in Table 1.

NQ320K. NQ320K is a popular dataset for evaluating generative retrieval models [42, 44]. It is based on the Natural Questions (NQ) dataset proposed by Google [24]. NQ320k consists of 320k query-document pairs, where the documents are gathered from Wikipedia pages, and the queries are natural language questions.

MS MARCO. MS MARCO is a collection of queries and web pages from Bing search. Similarly to NQ320k [60], we sample a subset of labeled documents and use their corresponding queries for training. We evaluate the models on the queries of the MS MARCO dev set and retrieval on the sampled document subset.

4.2 Baselines and Evaluation Metrics

Three types of baselines are considered in our work, including sparse retrieval methods, dense retrieval methods, and generative retrieval methods. More details about baselines can be found in Appendix A.1. On NQ320K, we use Recall@{1,5,10} and Mean Reciprocal Rank (MRR)@10 as evaluation metrics, following [32]. On MS MARCO, we use Recall@{1,5,10} and MRR@10.

4.3 Implementation Details

Models and Hyperparameters. In our experiments, we utilize LLaDA [34] and Dream [51] as the diffusion language models. As for the learnable DocIDs, the length of DocIDs is set as $l = 3$ and the size of i-th codebook is $K_i = 512, i \in \{1, 2, 3\}$. The maximum number of tokens is 12 for linguistic DocIDs. We optimize the model using AdamW and set the learning rate to $5e-4$. The batch size is 32, and the total number of training epochs is 30.

Data Augmentation. Following previous work [43, 44, 61], we use query generation models to generate synthetic (query, document) pairs for training data augmentation. Following Sun et al. [40], we use the pre-trained query generation model from DocT5Query [11] to augment the NQ320K and MS MARCO datasets. In query generation, we use nucleus sampling with parameters $p = 0.8, t = 0.8$ and generate ten queries for each document in the collection.

5 Results and Analysis

In this section, we present the experimental results to systematically evaluate the performance of DiffuGR and analyze the contribution of its core components. Our evaluation aims to address the following key research questions:

Q1 How does DiffuGR perform compared to baseline methods?
Q2 How does DiffuGR balance the quality–latency tradeoff?
Q3 How does DiffuGR generate top-k results without beam search?
Q4 How do hyperparameters affect DiffuGR performance, including decoding strategies and model scales?

5.1 Performance Comparison (Q1)

To answer **Q1**, we compare DiffuGR against various baselines across sparse, dense, and generative retrieval paradigms.

Results on NQ320K. Table 2 summarizes the performance comparison on the NQ320K dataset. We draw the following observations:

(1) The proposed DiffuGR (Dream-Linguistic) significantly outperforms all generative retrieval baselines, achieving 20.55% and 14.06% higher R@1 and MRR@10, respectively, compared to the best-performing baseline, DDRO [32]. This highlights the effectiveness of the diffusion process in accurate single-document identification.

(2) Different types of DocID affect the performance of DiffuGR. Overall, using linguistic DocIDs yields better performance than using learnable DocIDs. This is likely because linguistic DocIDs can directly leverage the semantic space of existing models, whereas learnable DocID require vocabulary expansion and semantic remapping, which further processes learning difficulty.

(3) Compared with R@1, our method achieves marginal improvement on R@5 and R@10 metrics. This is attributed to the fact that DiffuGR does not support beam search for top-k generation. We deeply investigate this issue in Section 5.3.

Results on MS MARCO. To further validate the scalability and robustness of the proposed DiffuGR, we conducted experiments on the more complex MS MARCO dataset. Table 3 presents the performance comparison on the MS MARCO dataset, where DiffuGR still achieves impressive performance on R@1. However, DiffuGR's performance on R@5 and R@10 is lower than that of some generative retrieval baselines. As for the issue already mentioned in the previous sections, this is related to the fact that the current diffusion language models do not support beam search-based top-k generation. In addition, we also observe that most methods perform worse on the MS MARCO dataset than on NQ320K. This may be because the MS MARCO dataset is composed of real user search logs, while the documents in NQ320K are from Wikipedia. Therefore, MS MARCO is more aligned with real-world search scenarios and poses greater challenges.

Table 2: Performance comparison on NQ320K. The best results are shown in bold. The second-best values are underlined. † indicates the result is significantly improved with paired t-test at $p < 0.05$ level. Abbreviations denote DocIDs used for GR baselines: SI – Semantic ID; PQ – Product Quantization; NG – N-grams; TU – Title + URL. Results of baselines are copied from [32]. R@k is short for Recall@k.

Model	R@1	R@5	R@10	MRR@10
Sparse retrieval				
BM25	14.06	36.91	47.93	23.60
DocT5Query	19.07	43.88	55.83	29.55
Dense retrieval				
DPR	22.78	53.44	68.58	35.92
ANCE	24.54	54.21	69.08	36.88
RepBERT	22.57	52.20	65.65	35.13
Sentence-T5	22.51	52.00	65.12	34.95
Generative retrieval				
DSI (SI)	27.42	47.26	56.58	34.31
DSI-QG (SI)	30.17	53.20	66.37	38.85
NCI (SI)	32.69	55.82	69.20	42.84
SEAL (NG)	29.30	54.12	68.53	40.34
Ultron (TU)	33.78	54.20	67.05	42.51
Ultron (PQ)	25.64	53.09	65.75	37.12
ROGER-NCI (SI)	33.20	56.34	69.80	43.45
ROGER-Ultron (TU)	35.90	55.59	69.86	44.92
MINDER (SI)	31.00	55.50	65.79	43.50
LTRGR (SI)	32.80	56.20	68.74	44.80
DDRO (TU)	40.86	53.12	55.98	45.99
DDRO (PQ)	48.92	64.10	67.31	55.51
Ours				
LLaDA-Learnable	62.63†	65.97†	66.01	64.20†
LLaDA-Linguistic	66.15†	66.83†	66.90	66.26†
Dream-Learnable	65.63†	68.23†	68.54†	66.62†
Dream-Linguistic	**69.47**†	**69.61**†	**69.78**†	**69.57**†

Table 3: Performance comparison on MS MARCO. The best results are shown in bold. The second-best values are underlined. † indicates the result is significantly improved with paired t-test at $p < 0.05$ level. Abbreviations denote DocIDs used for GR baselines: SI – Semantic ID; PQ – Product Quantization; NG – N-grams; TU – Title + URL. Results of baselines are copied from [32]. R@k is short for Recall@k.

Model	R@1	R@5	R@10	MRR@10
Sparse retrieval				
BM25	18.94	42.82	55.07	29.24
DocT5Query	23.27	49.38	63.61	34.81
Dense retrieval				
DPR	29.08	62.75	73.13	43.41
ANCE	29.65	63.43	74.28	44.09
RepBERT	25.25	58.41	69.18	38.48
Sentence-T5	27.27	58.91	72.15	40.69
Generative retrieval				
DSI (SI)	25.74	43.58	53.84	33.92
DSI-QG (SI)	28.82	50.74	62.26	38.45
NCI (SI)	29.54	57.99	67.28	40.46
SEAL (NG)	27.58	52.47	61.01	37.68
Ultron (TU)	29.82	60.39	68.31	42.53
Ultron (PQ)	31.55	63.98	73.14	45.35
ROGER-NCI	30.61	59.02	68.78	42.02
ROGER-Ultron	33.07	63.93	75.13	46.35
MINDER (SI)	29.98	58.37	71.92	42.51
LTRGR (SI)	32.69	64.37	72.43	47.85
DDRO (PQ)	32.92	64.36	73.02	45.76
DDRO (TU)	38.24	**66.46**	**74.01**	**50.07**
Ours				
LLaDA-Learnable	42.97†	43.58	43.60	43.24
LLaDA-Linguistic	**45.38**†	47.78	47.86	46.39
Dream-Learnable	43.60†	44.22	44.29	43.88
Dream-Linguistic	45.05†	47.05	47.21	45.96

5.2 Quality-Speed Tradeoff (Q2)

To answer **Q2**, we conduct an ablation study to evaluate the speed and quality with varying numbers of denoising iterations. Figure 3 shows the result on NQ320K. Results on MS MARCO show similar trends and can be found in Appendix A.2.1. As shown in Figure 3, increasing the number of iterations consistently improves retrieval performance, with higher R@1 scores achieved at larger iteration steps. However, the computational cost grows with the increase of iterations, leading to longer inference latency. For example, reducing the steps by half results in a noticeable decrease in latency while maintaining competitive retrieval accuracy, whereas further reducing the steps yields faster decoding at the expense of quality degradation. For comparison, we tested Qwen-2.5-7B R@1 and the number of samples processed per second. It can be observed that although Qwen achieved relatively good performance, its processing speed was relatively slow and failing to flexibly balance the tradeoff between performance and speed. These findings highlight a key advantage of DiffuGR, i.e., controllable generation process allows for dynamic adjustment of inference cost, making them suitable for scenarios with heterogeneous efficiency requirements. In particular, DiffuGR can leverage this flexibility to adaptively select iteration budgets depending on latency constraints of deployment.

5.3 Effect of Pseudo Beam Search (Q3)

To answer **Q3** and assess the effectiveness of the proposed pseudo beam search strategies, we conducted ablation experiments comparing three configurations of DiffuGR: (1) the vanilla model without pseudo beam search, (2) the model equipped with query augmentation, and (3) the model utilizing intermediate denoising candidates. All methods share identical architectures and training settings to ensure fair comparison.

Table 4 reports the Recall@{1,10,100} on NQ320K. Results on MS MARCO can be found in Appendix A.2.2. The results demonstrate that both strategies improve retrieval accuracy over the vanilla baseline. Specifically, query augmentation yields consistent gains by introducing diversity in query representations, allowing the model to explore multiple latent retrieval hypotheses.

The intermediate denoising approach also improves recall metrics, albeit to a lesser extent, as it relies on stochastic variations

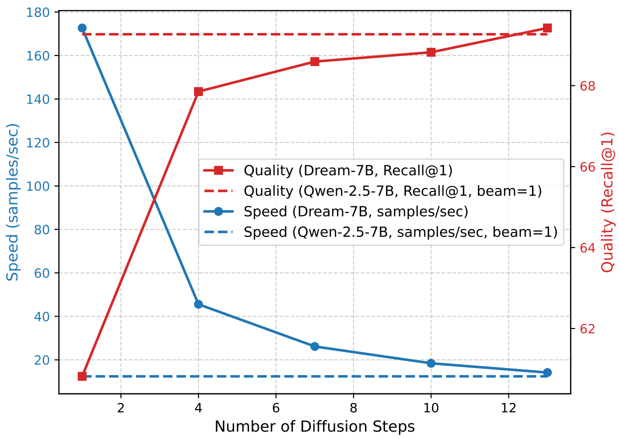

Figure 3: Quality-speed comparison on NQ320K for Dream 7B and Qwen2.5-7B. By adjusting the denoising steps, the performance of DiffuGR can be flexibly tuned towards either speed or quality.

Table 4: Effect of pseudo beam search on NQ320K. The best results are shown in bold. † indicates the result is significantly improved with paired t-test at $p < 0.05$ level. R@k is short for Recall@k.

Method	R@1	R@10	R@100
LLaDA-Linguistic	66.15	66.90	66.90
+ query augmentation	**66.62**†	66.96†	66.96†
+ intermediate denoising states	66.57†	**67.34**†	**67.34**†
Dream-Linguistic	69.47	69.78	69.78
+ query augmentation	69.50	69.82†	69.82†
+ intermediate denoising states	**69.57**†	**69.83**†	**69.83**†

Table 5: Impact of denoising strategies on NQ320K. The best results are shown in bold. † indicates the result is significantly improved with paired t-test at $p < 0.05$ level. ‡ denotes the default setting. R@k is short for Recall@k.

Method	R@1	R@10	R@100
Dream-Linguistic			
+ random	67.81	68.12	68.12
+ maskgit plus‡	**69.47**†	**69.78**†	**69.78**†
+ topk margin	69.38	69.69	69.69
+ entropy	69.02	69.33	69.33

within the denoising process rather than semantic diversity. Although both methods have improved the performance of DiffuGR, the performance improvement is limited compared with the beam search technique under the auto-regressive model. Therefore, developing more effective methods to enable diffusion-based top-k generation is an important direction for further improvement.

5.4 Impact of Hyperparameters (Q4)

To answer **Q4**, experiments are conducted to investigate the effect of denoising strategies and model scales.

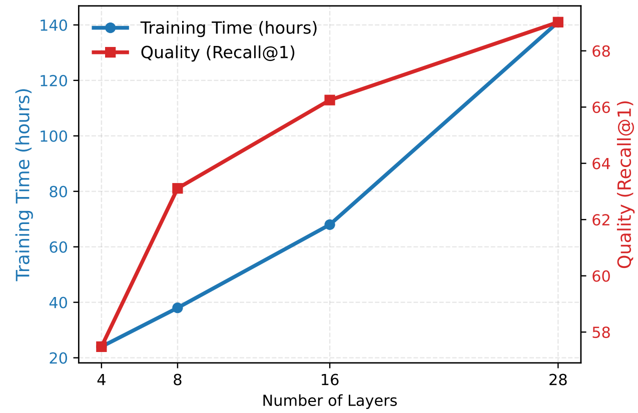

Figure 4: Retrieval performance of DiffuGR on NQ320K under different model scales. Larger models consistently yield higher retrieval accuracy, demonstrating that increased model sizes enhance the retrieval process. However, larger models also incur substantially higher training costs, revealing a clear tradeoff between effectiveness and efficiency.

5.4.1 Impact of Denoising Strategies. Table 5 shows the impact of denoising strategies on NQ320K. Results on MS MARCO can be found in Appendix A.2.3. As shown in Table 5, different denoising strategies yield distinct retrieval performance under the same testing configuration. We observe that the *maskgit plus* and *topk margin* strategies consistently outperform the random baseline, demonstrating the benefit of confidence-guided denoising in reducing uncertainty accumulation during generation. The *entropy* strategy achieves comparable performance, suggesting that both local certainty (margin) and global uncertainty (entropy) are effective cues for progressive denoising. These findings indicate that token generation scheduling plays a crucial role in improving retrieval accuracy and stability of DiffuGR.

5.4.2 Impact of Model Scales. To investigate how model scales influence the performance of DiffuGR, we conducted a series of experiments using diffusion language models with varying parameter scales. Specifically, we experiment with Dream of different model depths, comprising 4, 8, and 16 transformer layers, derived from the original 28-layer Dream architecture. All models are trained until no further performance improvement is observed. To ensure fair comparisons, we keep all hyperparameters identical across models, including optimizer settings, learning rate schedule, and GPU configuration. Moreover, the smaller models are initialized from the corresponding layers of the pretrained Dream model; that is, the parameters of the first n layers are directly copied from the full 28-layer checkpoint. This strategy provides the smaller models with a strong initialization, enabling them to achieve better performance. Figure 4 presents the results on NQ320K. We can observe that larger models consistently outperform smaller ones on Recall@1, demonstrating that increasing model scales enhances the diffusion-based retrieval process. However, we also observe a significant increase in training time as model size grows, which means the training consumes more costs. In particular, the 16-layer model requires nearly twice the training resources of the 8-layer counterpart, reflecting a clear tradeoff between performance and efficiency. Overall, these

Table 6: Perform comparison between diffusion-based generation and auto-regressive generation on the NQ320K dataset. R@k is short for Recall@k. Best results are shown in bold. † indicates the result is significantly improved with paired t-test at $p < 0.05$ level.

Method	R@1	R@10	R@100
Qwen2.5-7B-Linguistic	68.96	69.27	69.27
Dream-Learnable	65.63	68.54	68.54
Dream-Linguistic	**69.47**†	**69.78**†	**69.78**†

findings indicates that moderately scaled models may strike a better balance between effectiveness and efficiency, making them more practical under limited resources.

Besides, we also conducted experiments to assess the effectiveness of diffusion language models compared to state-of-the-art auto-regressive large language models under the same model size. Specifically, we benchmark our diffusion-based model against Qwen2.5-7B, a representative 7B-parameter auto-regressive model known for its strong performance across a variety of language understanding and generation tasks. This setup ensures that performance differences can be attributed to the generation paradigms rather than model scales. As shown in Table 6, our diffusion model achieves better retrieval accuracy compared to that of Qwen2.5-7B. Such results further verify the effectiveness of diffusion-based generation.

6 Conclusion

In this paper, we have proposed a generative document retrieval framework based on diffusion language models, dubbed DiffuGR, and have investigated its performance on two widely used datasets. Our results show that DiffuGR achieves competitive performance compared with strong auto-regressive baselines, while its generation paradigm is more suitable for generative document retrieval tasks. Moreover, the generation quality and speed of DocIDs can be flexibly controlled by adjusting the number of denoising iterations. However, DiffuGR still face several challenges: (1) DiffuGR does not support beam search, resulting in only marginal improvements in larger top-k generation; (2) DiffuGR does not support constrained decoding, which may lead to invalidly generated DocIDs.

In future work, we plan to further improve the performance of diffusion language models for generative retrieval, aiming to address the above existing challenges. Besides, we are also interested in investigating whether larger diffusion models trained on more query-document pairs can further improve retrieval performance.

References

[1] Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. 2025. Block Diffusion: Interpolating Between Autoregressive and Diffusion Language Models. arXiv:2503.09573 [cs.LG] https://arxiv.org/abs/2503.09573

[2] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. 2021. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems* 34 (2021), 17981–17993.

[3] Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Wen tau Yih, Sebastian Riedel, and Fabio Petroni. 2022. Autoregressive Search Engines: Generating Substrings as Document Identifiers. In *NeurIPS*.

[4] Hongru Cai, Yongqi Li, Ruifeng Yuan, Wenjie Wang, Zhen Zhang, Wenjie Li, and Tat-Seng Chua. 2025. Exploring training and inference scaling laws in generative retrieval. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1339–1349.

[5] Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. 2022. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems* 35 (2022), 28266–28279.

[6] Daniel Fernando Campos, Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, Li Deng, and Bhaskar Mitra. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *ArXiv* abs/1611.09268 (2016).

[7] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive Entity Retrieval. In *ICLR*.

[8] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. 2022. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 11315–11325.

[9] Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Yiqun Liu, Yixing Fan, and Xueqi Cheng. 2023. A Unified Generative Retriever for Knowledge-Intensive Language Tasks via Prompt Learning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*, Hsin-Hsi Chen, Wei-Jou (Edward) Duh, Hen-Hsen Huang, Makoto P. Kato, Josiane Mothe, and Barbara Poblete (Eds.). ACM, 1448–1457. doi:10.1145/3539618.3591631

[10] Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Y. Liu, Yixing Fan, and Xueqi Cheng. 2022. CorpusBrain: Pre-train a Generative Retrieval Model for Knowledge-Intensive Language Tasks. In *CIKM*.

[11] David R. Cheriton. 2019. From Doc2query to DocTTTTTquery. *Online preprint*.

[12] Nicola De Cao, Ledell Wu, Kashyap Popat, Mikel Artetxe, Naman Goyal, Mikhail Plekhanov, Luke Zettlemoyer, Nicola Cancedda, Sebastian Riedel, and Fabio Petroni. 2022. Multilingual Autoregressive Entity Linking. *Transactions of the Association for Computational Linguistics* 10 (2022), 274–290. doi:10.1162/tacl_a_00460

[13] Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[14] Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, Hao Peng, and Lingpeng Kong. 2025. Scaling Diffusion Language Models via Adaptation from Autoregressive Models. *International Conference on Learning Representations* (2025).

[15] Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. 2023. DiffuSeq: Sequence to Sequence Text Generation with Diffusion Models. In *International Conference on Learning Representations, ICLR*.

[16] Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. 2023. DiffuSeq-v2: Bridging Discrete and Continuous Text Spaces for Accelerated Seq2Seq Diffusion Models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, 9868–9875.

[17] Ishaan Gulrajani and Tatsunori Hashimoto. 2023. Likelihood-Based Diffusion Language Models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

[18] Zhengfu He, Tianxiang Sun, Qiong Tang, Kuanning Wang, Xuanjing Huang, and Xipeng Qiu. 2023. DiffusionBERT: Improving Generative Masked Language Models with Diffusion Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 4521–4534. doi:10.18653/v1/2023.acl-long.248

[19] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems* 33 (2020), 6840–6851.

[20] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. 2021. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in neural information processing systems* 34 (2021), 12454–12465.

[21] Inception Labs. 2025. Mercury: Ultra-Fast Language Models Based on Diffusion. https://www.inceptionlabs.ai/introducing-mercury. Accessed: 2025-06-16.

[22] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Yu Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *EMNLP*.

[23] Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham Kakade, and Sitan Chen. 2025. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. *arXiv preprint arXiv:2502.06768* (2025).

[24] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc V. Le, and Slav Petrov. 2019. Natural Questions: A Benchmark for Question Answering Research. *TACL* 7 (2019), 453–466.

[25] Xiaoxi Li, Yujia Zhou, and Zhicheng Dou. 2024. UniGen: A Unified Generative Framework for Retrieval and Question Answering with Large Language Models. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (Eds.). AAAI Press, 8688–8696. doi:10.1609/AAAI.

V38I8.28714
[26] Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B Hashimoto. 2022. Diffusion-LM Improves Controllable Text Generation. In *Conference on Neural Information Processing Systems, NeurIPS*.
[27] Yongqi Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. 2023. Multiview identifiers enhanced generative retrieval. *arXiv preprint arXiv:2305.16675* (2023).
[28] Yongqi Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. 2024. Learning to rank in generative retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 8716–8723.
[29] Zhenghao Lin, Yeyun Gong, Yelong Shen, Tong Wu, Zhihao Fan, Chen Lin, Nan Duan, and Weizhu Chen. 2023. Text generation with diffusion language models: a pre-training approach with continuous paragraph denoise. In *Proceedings of the 40th International Conference on Machine Learning* (Honolulu, Hawaii, USA) *(ICML'23)*. JMLR.org, Article 867, 14 pages.
[30] Aaron Lou, Chenlin Meng, and Stefano Ermon. 2024. Discrete Diffusion Language Modeling by Estimating the Ratios of the Data Distribution. In *International Conference on Machine Learning, ICML*.
[31] Sanket Vaibhav Mehta, Jai Gupta, Yi Tay, Mostafa Dehghani, Vinh Quang Tran, Jinfeng Rao, Marc-Alexander Najork, Emma Strubell, and Donald Metzler. 2022. DSI++: Updating Transformer Memory with New Documents. *ArXiv* abs/2212.09744 (2022).
[32] Kidist Amde Mekonnen, Yubao Tang, and Maarten de Rijke. 2025. Lightweight and Direct Document Relevance Optimization for Generative Information Retrieval. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1327–1338.
[33] Jianmo Ni, Gustavo Hern'andez 'Abrego, Noah Constant, Ji Ma, Keith B. Hall, Daniel Matthew Cer, and Yinfei Yang. 2022. Sentence-T5: Scalable Sentence Encoders from Pre-trained Text-to-Text Models. In *Findings of ACL*.
[34] Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025. Large language diffusion models. *arXiv preprint arXiv:2502.09992* (2025).
[35] Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. 2024. Your Absorbing Discrete Diffusion Secretly Models the Conditional Distributions of Clean Data. *arXiv preprint arXiv:2406.03736* (2024).
[36] Ronak Pradeep, Kai Hui, Jai Gupta, Adam D Lelkes, Honglei Zhuang, Jimmy Lin, Donald Metzler, and Vinh Q Tran. 2023. How does generative retrieval scale to millions of passages? *arXiv preprint arXiv:2305.11841* (2023).
[37] Shanbao Qiao, Xuebing Liu, and Seung-Hoon Na. 2023. DiffusionRet: Diffusion-enhanced generative retriever using constrained decoding. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. 9515–9529.
[38] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan H. Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Maheswaran Sathiamoorthy. 2023. Recommender Systems with Generative Retrieval. arXiv:2305.05065 [cs.IR]
[39] Jiaming Song, Chenlin Meng, and Stefano Ermon. 2021. Denoising Diffusion Implicit Models. In *Proc. of ICLR*. OpenReview.net.
[40] Weiwei Sun, Lingyong Yan, Zheng Chen, Shuaiqiang Wang, Haichao Zhu, Pengjie Ren, Zhumin Chen, Dawei Yin, Maarten Rijke, and Zhaochun Ren. 2023. Learning to tokenize for generative retrieval. *Advances in Neural Information Processing Systems* 36 (2023), 46345–46361.
[41] Yubao Tang, Ruqing Zhang, Jiafeng Guo, Jiangui Chen, Zuowei Zhu, Shuaiqiang Wang, Dawei Yin, and Xueqi Cheng. 2023. Semantic-Enhanced Differentiable Search Index Inspired by Learning Strategies. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, Ambuj K. Singh, Yizhou Sun, Leman Akoglu, Dimitrios Gunopulos, Xifeng Yan, Ravi Kumar, Fatma Ozcan, and Jieping Ye (Eds.). ACM, 4904–4913. doi:10.1145/3580305.3599903
[42] Yi Tay, Vinh Quang Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. 2022. Transformer Memory as a Differentiable Search Index. In *NeurIPS*.
[43] Kexin Wang, Nandan Thakur, Nils Reimers, and Iryna Gurevych. 2022. GPL: Generative Pseudo Labeling for Unsupervised Domain Adaptation of Dense Retrieval. In *NAACL*.
[44] Yujing Wang, Ying Hou, Hong Wang, Ziming Miao, Shibin Wu, Hao Sun, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, Zheng Liu, Xing Xie, Hao Sun, Weiwei Deng, Qi Zhang, and Mao Yang. 2022. A Neural Corpus Indexer for Document Retrieval. In *NeurIPS*.
[45] Shiguang Wu, Zhaochun Ren, Xin Xin, Jiyuan Yang, Mengqi Zhang, Zhumin Chen, Maarten de Rijke, and Pengjie Ren. 2025. Constrained Auto-Regressive Decoding Constrains Generative Retrieval. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2429–2440.
[46] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *ICLR*.
[47] Tianchi Yang, Minghui Song, Zihan Zhang, Haizhen Huang, Weiwei Deng, Feng Sun, and Qi Zhang. 2023. Auto Search Indexer for End-to-End Document Retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, 6955–6970. https://aclanthology.org/2023.findings-emnlp.464
[48] Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. Beyond autoregression: Discrete diffusion for complex reasoning and planning. *International Conference on Learning Representations* (2025).
[49] Jiacheng Ye, Shansan Gong, Liheng Chen, Lin Zheng, Jiahui Gao, Han Shi, Chuan Wu, Xin Jiang, Zhenguo Li, Wei Bi, et al. 2024. Diffusion of thought: Chain-of-thought reasoning in diffusion language models. *Advances in Neural Information Processing Systems* 37 (2024), 105345–105374.
[50] Jiacheng Ye, Zhenyu Wu, Jiahui Gao, Zhiyong Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. Implicit search via discrete diffusion: A study on chess. *International Conference on Learning Representations* (2025).
[51] Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487* (2025).
[52] Jiasheng Ye, Zaixiang Zheng, Yu Bao, Lihua Qian, and Quanquan Gu. 2025. Diffusion Language Models Can Perform Many Tasks with Scaling and Instruction-Finetuning. arXiv:2308.12219 [cs.CL] https://arxiv.org/abs/2308.12219
[53] Hansi Zeng, Chen Luo, Bowen Jin, Sheikh Muhammad Sarwar, Tianxin Wei, and Hamed Zamani. 2023. Scalable and Effective Generative Information Retrieval. *CoRR* abs/2311.09134 (2023). arXiv:2311.09134 doi:10.48550/ARXIV.2311.09134
[54] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2020. Repbert: Contextualized text embeddings for first-stage retrieval. *arXiv preprint arXiv:2006.15498* (2020).
[55] Yidan Zhang, Ting Zhang, Dong Chen, Yujing Wang, Qi Chen, Xing Xie, Hao Sun, Weiwei Deng, Qi Zhang, Fan Yang, Mao Yang, Qingmin Liao, and Baining Guo. 2023. IRGen: Generative Modeling for Image Retrieval. arXiv:2303.10126 [cs.CV]
[56] Zhen Zhang, Xinyu Ma, Weiwei Sun, Pengjie Ren, Zhumin Chen, Shuaiqiang Wang, Dawei Yin, Maarten de Rijke, and Zhaochun Ren. 2025. Replication and Exploration of Generative Retrieval over Dynamic Corpora. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 3325–3334.
[57] Jujia Zhao, Wenjie Wang, Chen Xu, Xiuying Chen, Zhaochun Ren, and Suzan Verberne. 2025. Unifying Search and Recommendation: A Generative Paradigm Inspired by Information Theory. *arXiv preprint arXiv:2504.06714* (2025).
[58] Yujia Zhou, Zhicheng Dou, and Ji-Rong Wen. 2023. Enhancing generative retrieval with reinforcement learning from relevance feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 12481–12490.
[59] Yujia Zhou, Jing Yao, Zhicheng Dou, Yiteng Tu, Ledell Wu, Tat-Seng Chua, and Ji-Rong Wen. 2024. ROGER: Ranking-Oriented Generative Retrieval. *ACM Trans. Inf. Syst.* 42, 6, Article 155 (Oct. 2024), 25 pages. doi:10.1145/3603167
[60] Yujia Zhou, Jing Yao, Zhicheng Dou, Ledell Yu Wu, Peitian Zhang, and Ji rong Wen. 2022. Ultron: An Ultimate Retriever on Corpus with a Model-based Indexer. *ArXiv* abs/2208.09257 (2022).
[61] Shengyao Zhuang, Houxing Ren, Linjun Shou, Jian Pei, Ming Gong, G. Zuccon, and Daxin Jiang. 2022. Bridging the Gap Between Indexing and Retrieval for Differentiable Search Index with Query Generation. *ArXiv* abs/2206.10128 (2022).

A Appendix

A.1 Baselines

The sparse retrieval baselines are as follows:

- **BM25** uses the tf-idf feature to measure term weights; we use the implementation from http://pyserini.io/.
- **DocT5Query** expands a document with possible queries predicted by a finetuned T5 with this document as the input.

The dense retrieval baselines are as follows:

- **DPR** [22], a dual-encoder model using the representation of the `[CLS]` token of BERT.
- **ANCE** [46], an asynchronously updated ANN indexer is utilized to mine hard negatives for training a RoBERTa-based dual-encoder model.
- **Sentence-T5** [33], a dual-encoder model that uses T5 to produce continuous sentence embeddings.

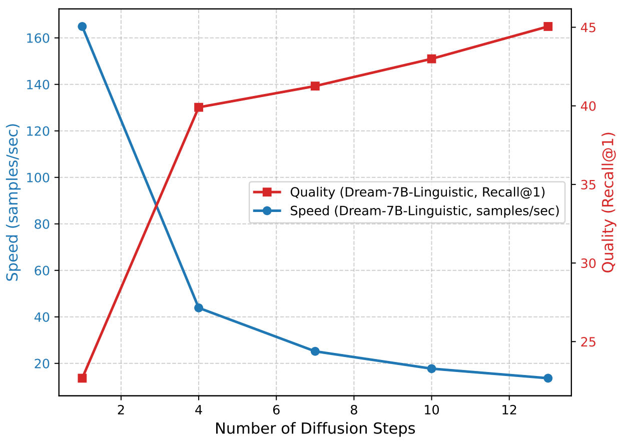

Figure 5: Quality-speed tradeoff on MS MARCO.

- **RepBERT** [54], a BERT-based model that generates fixed-length contextualized embeddings with query-document relevance computed via inner product similarity.

The generative retrieval baselines are as follows:

- **DSI** [42], which represents documents using hierarchical K-means clustering results, and indexes documents using the first 32 tokens as pseudo-queries. As the original code is not open source, we refer to the results reported by [44].
- **SEAL** [3] uses arbitrary n-grams in documents as DocIDs, and retrieves documents under the constraint of a pre-built FM-indexer. We refer to the results reported by [44].
- **DSI-QG** [61] uses a query generation model to augment the document collection. We refer to the results reported by [44].
- **NCI** [44] uses a prefix-aware weight-adaptive decoder and various query generation strategies, including DocAsQuery and DocT5Query. In particular, NCI augments training data by generating 15 queries for each document.
- **Ultron** [60] uses a three-stage training pipeline and represents documents through three types of identifiers.
- **ROGER** [59],which transfers knowledge from a dense retriever to a generative retriever via knowledge distillation.
- **MINDER** [27],which assigns multiple identifiers, including titles, n-grams, and synthetic queries, to documents and pairs them for indexing.
- **LTRGR** [28],which trains on pairwise relevance objectives using margin-based ranking loss for optimization.
- **GenRRL** [58],which incorporates pointwise, pairwise, and listwise relevance optimization through reinforcement learning, using document summaries and URLs as docids.
- **DDRO** [32], which enhances generative retrieval by directly aligning generation with document-level relevance estimation.

A.2 Additional Results

A.2.1 Quality-Speed Tradeoff on MS MARCO. The results of quality-speed tradeoff on MS MARCO is shown in Figure 5.

A.2.2 Effect of Pseudo Beam Search on MS MARCO. The results of effect of pseudo beam search on MS MARCO is shown in Table 7.

Table 7: Effect of pseudo beam search on MS MARCO. The best results are shown in bold. † indicates the result is significantly improved with paired t-test at $p < 0.05$ level. R@k is short for Recall@k.

Method	R@1	R@10	R@100
LLaDA-Learnable	42.97	43.60	43.60
+ query augmentation	43.21†	43.68†	43.71†
+ intermediate denoising states	**43.59**†	**43.89**†	**43.89**†
Dream-Learnable	43.60	44.29	44.29
+ query augmentation	**44.22**†	44.45†	44.60†
+ intermediate denoising states	44.20†	**44.50**†	**44.61**†

Table 8: Impact of denoising strategies on MS MARCO. The best results are shown in bold. † indicates the result is significantly improved with paired t-test at $p < 0.05$ level. ‡ denotes the default setting. R@k is short for Recall@k.

Method	R@1	R@10	R@100
Dream-Linguistic			
+ random	39.77	41.76	42.04
+ maskgit plus‡	**45.05**†	47.13	47.21
+ topk margin	44.97	**47.21**	**47.52**
+ entropy	44.09	46.25	46.56

A.2.3 Impact of Denoising Strategies on MS MARCO. The results of the impact of denoising strategies on MS MARCO is shown in Table 8

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009