Listwise Generative Retrieval Models via a Sequential Learning Process

YUBAO TANG, RUQING ZHANG*, and JIAFENG GUO†, Institute of Computing Technology, Chinese Academy of Sciences; University of Chinese Academy of Sciences, Beijing, China
MAARTEN DE RIJKE, University of Amsterdam, Amsterdam, The Netherlands
WEI CHEN and XUEQI CHENG, Institute of Computing Technology, Chinese Academy of Sciences; University of Chinese Academy of Sciences, Beijing, China

Recently, a novel generative retrieval (GR) paradigm has been proposed, where a single sequence-to-sequence model is learned to directly generate a list of relevant document identifiers (docids) given a query. Existing GR models commonly employ maximum likelihood estimation (MLE) for optimization: this involves maximizing the likelihood of a single relevant docid given an input query, with the assumption that the likelihood for each docid is independent of the other docids in the list. We refer to these models as the pointwise approach in this paper. While the pointwise approach has been shown to be effective in the context of GR, it is considered sub-optimal due to its disregard for the fundamental principle that ranking involves making predictions about lists. In this paper, we address this limitation by introducing an alternative listwise approach, which empowers the GR model to optimize the relevance at the docid list level. Specifically, we view the generation of a ranked docid list as a sequence learning process: at each step we learn a subset of parameters that maximizes the corresponding generation likelihood of the i-th docid given the (preceding) top $i-1$ docids. To formalize the sequence learning process, we design a positional conditional probability for GR. To alleviate the potential impact of beam search on the generation quality during inference, we perform relevance calibration on the generation likelihood of model-generated docids according to relevance grades. We conduct extensive experiments on representative binary and multi-graded relevance datasets. Our empirical results demonstrate that our method outperforms state-of-the-art GR baselines in terms of retrieval performance.

CCS Concepts: • **Information systems** → **Retrieval models and ranking**.

Additional Key Words and Phrases: Document retrieval, Generative retrieval, Listwise approach

1 INTRODUCTION

Document retrieval plays a critical role in many information retrieval (IR) related tasks, e.g., web search [19, 70] and question answering [33, 80]. It aims to return an initial set of potentially relevant documents from a large-scale document repository when given a query. Recently, a new retrieval paradigm called *generative retrieval* (GR) [69] for document retrieval has been proposed. The key idea is to fully parameterize different components of indexing and retrieval within a single consolidated model, in which the information of all the documents in a corpus is encoded into the model parameters. In essence, this paradigm formalizes the document retrieval

*Research conducted when the author was at the University of Amsterdam.
†Jiafeng Guo is the corresponding author.

Authors' addresses: Yubao Tang, tangyubao@ict.ac.cn; Ruqing Zhang, zhangruqing@ict.ac.cn; Jiafeng Guo, guojiafeng@ict.ac.cn, Institute of Computing Technology, Chinese Academy of Sciences; University of Chinese Academy of Sciences, Beijing, No. 6 Kexueyuan South Road, Haidian District, China, 100190; Maarten de Rijke, m.derijke@uva.nl, University of Amsterdam, Amsterdam, The Netherlands; Wei Chen, chenwei@ict.ac.cn; Xueqi Cheng, cxq@ict.ac.cn, Institute of Computing Technology, Chinese Academy of Sciences; University of Chinese Academy of Sciences, Beijing, No. 6 Kexueyuan South Road, Haidian District, China, 100190.

ACM 1046-8188/2024/3-ART
https://doi.org/10.1145/3653712

task as a sequence-to-sequence (Seq2Seq) problem that directly maps string queries to relevant document identifiers (docids). Following the initial publication by Metzler et al. [69], many subsequent investigations [10, 15, 22, 84, 87, 105] have showcased the potential of this novel paradigm. In comparison to traditional dense retrieval [9, 63, 71, 99], GR has several advantages:

(i) During training, such a consolidated model can be optimized directly in an end-to-end manner towards a global objective. By generating docids token-by-token in an autoregressive fashion and conditioning them on the query, we can capture fine-grained interactions between the query and the document.
(ii) During inference, the need for a complicated explicit index structure is eliminated. Instead, docid generation is performed using a vocabulary with tens of thousands of words, aligned with identifiers of all the documents in the corpus. Such autoregressive decoding significantly reduces the memory space and computational costs.

The majority of existing GR models relies on the standard Seq2Seq objective, i.e., maximum likelihood estimation (MLE) [31, 51] with teacher forcing for learning. That is, during training, a number of queries are provided; each query is associated with a perfect ranked list of docids (in descending order of relevance scores); GR models operate in a pointwise manner. For example, as shown in Figure 1 (Top), existing works mainly focus on maximizing the likelihood of individual docids at a time. The final ranking is achieved by simply sorting the list based on the generated likelihood scores of these docids. In essence, the score assigned to each docid is independent of the other docids for a given query. This approach suffers from several issues: First, the learning objective under the MLE criterion is formalized as minimizing errors in generation of docids, rather than minimizing errors in rankings of docids, making it inconsistent with evaluation metrics like nDCG [38]. Second, given a query, the assumption that the query-docid pairs are generated independently and identically distributed (i.i.d.) is a strong assumption. Thirdly, the number of query-docid pairs can vary greatly from one query to another, leading to a GR model that is biased towards queries with a larger number of docid pairs [11].

In this paper, we design a novel listwise approach to GR, in which *docid lists* instead of *individual docids* are used as instances in learning, as shown in Figure 1 (Bottom). Inspired by listwise learning-to-rank [11, 48, 91], it is crucial to effectively capture the difference between a ranked list of docids produced by a GR model and the ranked list given as the ground truth. To formalize the listwise loss function for GR, our key idea is to view the problem of generating a ranked list of relevant docids as a sequential learning process: in each step we target to maximize the corresponding stepwise probability distribution. Specifically, at step 1, we aim to maximize the probability distribution that the top-1 docid is generated. At step $i > 1$, we maximize the i-th probability distribution given the top $i - 1$ docids. Leveraging the characteristics of GR, we define the probability distribution as the output sequence likelihood of generating each docid, token-by-token in an autoregressive fashion, and conditioned on the given query. To solve the sequential learning problem, we transform it into a single-objective optimization problem via linear scalarization, in which the position importance in ranking is highlighted [48]. By assigning appropriate weights to different ranking positions, the final listwise loss function can effectively emphasize the significance of each position and optimize the overall objective accordingly. The comparison between previous pointwise approaches and our proposed listwise approach for GR is illustrated in Figure 1. We refer to the GR model using the listwise loss function as ListGR.

At inference time, the trained ListGR model uses beam search to generate a ranked list of potentially-relevant docids, which are based on possibly erroneous previous steps. However, in the proposed listwise loss function, the predictive probability of each reference docid is maximized given the gold sub-sequence before it. To solve this decoding inconsistency problem, we propose to perform relevance calibration to re-train the model with a relevance calibration objective. This objective aims to calibrate the likelihood of generated candidate docids to better align with ground-truth ranked lists according to their relevance grades to the query.

Our main contributions are the following:

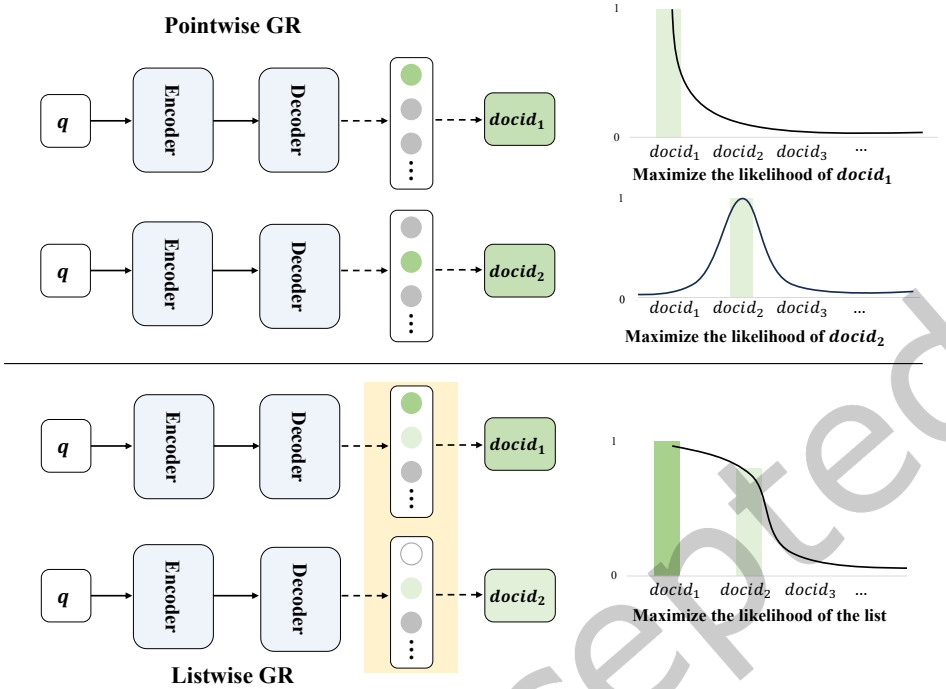

Fig. 1. Optimization objectives. Assume that the following are given: a query q and two ground-truth docids, $docid_1$ and $docid_2$, where $docid_1$ is more relevant than $docid_2$ to q. Top: Most existing GR work relies on maximum likelihood estimation, by maximizing the likelihood of the target docid for each query-docid pair. All relevant docids $docid_1$ and $docid_2$ are treated equally, sharing similar likelihood values. Bottom: A listwise objective (yellow rectangle) is designed for GR, directly modeling the ranked docid lists and incorporating positional information between $docid_1$ and $docid_2$ ($docid_1$ with darker green has a larger positional weight), resulting in a positional weighted likelihood.

(i) To the best of our knowledge, this is the first proposal for a listwise approach specifically designed for GR.
(ii) We formulate a listwise learning objective for GR, by directly minimizing the expected loss defined on the predicted docid list and the ground-truth list, and taking into account position information.
(iii) Our experimental results on five representative retrieval datasets demonstrate the effectiveness of our method, particularly on datasets with multi-graded relevance. Compared to the current state-of-the-art pointwise GR method, NCI, our approach achieves a significant improvement of 15.8% in terms of nDCG@5 on the ClueWeb 200K dataset.

The remainder of the paper is structured as follows. Section 2 introduces preliminary concepts necessary for understanding the proposed method. Section 3 outlines the details of our proposed method. Section 4 describes the experimental setup. Section 5 presents the experimental results and analysis, highlighting the performance of our method compared to existing approaches. Section 6 presents an overview of related work in the field. Finally, Section 7 provides a summary of the paper and discusses limitations and potential future research directions.

2 PRELIMINARIES

We first recall the basic idea of the GR paradigm and of listwise algorithms that have been widely adopted in learning-to-rank; Table 1 lists the most important notations used in the paper.

Table 1. Important notation.

D	Document set
d	Document in D
I_D	Docid set corresponding to D
id	Docid in the docid set I_D
$id^{(i)}$	i-th ranked docid in a ranked docid list
π_q	Ground-truth docid list of q
π_Q	Set of ground-truth docid lists of Q
Q	Query set
q	Query
g_θ	GR model parameters
w_t	Token in the docid id
x	Document set to be further ranked for a query
π_y	Ground-truth permutation of documents x for a query
y_i	Ranking position of document d_i in π_y
$y^{-1}(i)$	Index identifier of documents in the i-th position of π_y
h_ψ	Learning to rank function

2.1 Generative retrieval

Generative retrieval (GR) aims to directly generate a ranked list of docids for a given query using a text-to-text model. In the following, we summarize the model architecture, training, and inference process of GR.

2.1.1 ***Model architecture.*** In existing approaches, the GR model, represented as g_θ, usually makes use of a transformer-based encoder-decoder architecture to answer queries. The encoder is responsible for processing the input sequence, i.e., query or document, and extracting meaningful representations to capture the essential topics. Based on the representation produced by the encoder, the decoder is responsible for generating the target docid.

2.1.2 ***Document identifiers (docids).*** Tay et al. [84] propose two primary document identifiers to represent documents:

(i) Arbitrary unique integers without explicit semantic connections to the corresponding documents [84].
(ii) Structured semantic numbers that carry semantic associations with the documents, often obtained through techniques like hierarchical k-means clustering [84, 87].

Incorporating semantic associations between docids and documents improves the retrieval process [10, 22, 84, 87]. In this work, we adopt the structured semantic numbers for docid representation and we leave a detail discussion of the docid generation process to Section 4.5. Recently, alternative forms of docids such as n-grams and titles have been proposed. A comprehensive explanation of these docids can be found in Section 6.

2.1.3 ***Training and optimization.*** Maximum likelihood estimation (MLE) is widely employed in current GR methods to optimize two main tasks, i.e., the indexing task and the retrieval task, via maximizing the likelihood estimation of the target docid, given a document or query.

Indexing task. To memorize the corpus, the GR model g_θ takes the document d in the document set D as the input, and outputs its corresponding docid id in the docid set I_D with MLE optimization algorithm, defined as,

$$\mathcal{L}_{Indexing}(D, I_D; g_\theta) = -\sum_{d \in D} \log P(id \mid d; g_\theta), \tag{1}$$

where $P(id \mid d; g_\theta)$ is the likelihood of generation docid id,

$$P(id \mid d; g_\theta) = \prod_{t \in [1, |id|]} P(w_t \mid d, w_{<t}; g_\theta), \tag{2}$$

where w_t is the i-th ground-truth token in the id, and $w_{<t}$ represents the tokens before the i-th one in the id.

Retrieval task. A query q in the query set Q can have one or multiple associated docids, and these docids may possess varying degrees of relevance. For q, it has a ground-truth docid list, $\pi_q = [id^{(1)}, id^{(2)}, \ldots]$, in descending order of relevance, where $id^{(1)}$ is the docid ranked at the first position, and $id^{(2)}$ is the docid ranked at the second position, and so on. We denote the set of relevant docids for all the queries Q as π_Q. Relevance grades for documents are non-negative integers. A relevance grade of 0 indicates that the document is irrelevant to the query. The higher the integer value, the greater the relevance of the document to the given query. And $M(d)$ denotes the relevance grade of the document d to a query. To achieve the retrieval task effectively, the GR model also leverages MLE to learn how to map the query q in the query set Q to relevant docids, defined as,

$$\mathcal{L}_{Retrieval}(Q, \pi_Q; g_\theta) = -\sum_{q \in Q, id \in \pi_q} \log P(id \mid q; g_\theta), \tag{3}$$

where $P(id \mid q; g_\theta)$ is similar to Eq. (2), defined as

$$P(id \mid q; g_\theta) = \prod_{t \in [1, |id|]} P(w_t \mid q, w_{<t}; g_\theta). \tag{4}$$

Finally, the total loss incurred during training a GR model is a combination of the indexing loss and the retrieval loss, i.e.,

$$\mathcal{L}_{Total}(Q, D, I_D) = \mathcal{L}_{Indexing}(D, I_D; g_\theta) + \mathcal{L}_{Retrieval}(Q, \pi_Q; g_\theta). \tag{5}$$

2.1.4 ***Inference***. During inference, given a query, the GR model usually uses beam search [45] to generate the top-n ranked docids in an autoregressive manner, in descending order based on the conditional probability of each output. Note that, when generating the next token, the model relies on the former generated token, rather than the ground-truth token.

2.1.5 ***Discussion***. In current GR methods, MLE is primarily used to train query-docid pairs (as shown in Eq. (3)), which is a pointwise approach. This approach, however, is limited in its ability to support the model in generating the single most relevant docid even when a query has multiple relevant docids. During inference, the goal of the retrieval task is to obtain a ranked docid list, where the docids are ordered based on their relevance to the query. The pointwise approach fails to guarantee an optimal ordering of docids within the list.

To address this limitation and enhance the capability of the GR model to generate a high-quality ranked docid list, this work focuses on modeling and optimizing the relevance at the list level. By shifting the optimization objective from a pointwise perspective to a listwise perspective, we aim to further improve the overall effectiveness of the GR models.

2.2 ListMLE algorithm

In learning-to-rank (LTR), listwise approaches emphasize optimizing the entire ranked list of items for overall ranking performance. Listwise approaches recognize that the order in which items are presented in the list is crucial for accurate ranking. In the following, we describe a related algorithm for our work, including listMLE and position-aware listMLE.

2.2.1 ***ListMLE***. Formally, suppose $x = \{d_1, \ldots, d_n\} \in X$ is the subset of corpus D to be further ranked, obtained from an initial document retrieval step. And $\pi_y = [y_1, \ldots, y_n] \in Y$ is the corresponding ground-truth permutation of these documents, where y_i is the position of d_i, and $y^{-1}(i)$ is the index identifier of documents in the i-th position of π_y. Listwise LTR aims to learn a ranking function $h_\psi : X \to Y$, where ψ are the function parameters and H is the corresponding function space (i.e., $h \in H$), that can minimize the expected risk.

ListMLE [91] is a widely-used framework for listwise ranking that introduces a parameterized exponential probability distribution over all possible permutations, given the ranking function h_ψ. And it leverages negative log likelihood of the ground truth list as the loss function, defined as:

$$\mathcal{L}(x, \pi_y; h_\psi) = -\log P(\pi_y \mid x; h_\psi). \tag{6}$$

According to the Plackett-Luce model [64, 75], which is a distribution over permutations π_y, $P(\pi_y \mid x; h_\psi)$ can be defined as:

$$P(\pi_y \mid x; h_\psi) = \prod_{i=1}^{n} \frac{\exp(h_\psi(x_{y^{-1}(i)}))}{\sum_{k=i}^{n} \exp(h_\psi(x_{y^{-1}(k)}))}. \tag{7}$$

The probability of a list can be deconstructed into the product of stepwise conditional probabilities. Each i-th conditional probability represents the likelihood of a document being ranked at the i-th position, given that the preceding documents are ranked appropriately up to that point. i.e.,

$$P(\pi_y \mid x; h_\psi) = P(y^{-1}(1), \ldots, y^{-1}(n) \mid x; h_\psi) \tag{8}$$

$$= P(y^{-1}(1) \mid x; h_\psi) \prod_{i=2}^{n} P(y^{-1}(i) \mid x, y^{-1}(1), \ldots, y^{-1}(i-1); h_\psi), \tag{9}$$

where

$$P(y^{-1}(1) \mid x; h_\psi) = \frac{\exp(h_\psi(x_{y^{-1}(1)}))}{\sum_{k=1}^{n} \exp(h_\psi(x_{y^{-1}(k)}))}, \tag{10}$$

$$P(y^{-1}(i) \mid x, y^{-1}(1), y^{-1}(2), \ldots, y^{-1}(i-1); h_\psi) = \frac{\exp(h_\psi(x_{y^{-1}(i)}))}{\sum_{k=i}^{n} \exp(h_\psi(x_{y^{-1}(k)}))}, \forall i = 2, \ldots, n. \tag{11}$$

2.2.2 ***Position-aware ListMLE***. ListMLE, despite its effectiveness, ignores the significance of position importance [48]. Recognizing the impact of item positions for ranking, an advanced listwise ranking algorithm called *position-aware ListMLE* [p-ListMLE, 48] has been developed to take into account position information. p-ListMLE considers the ranking process as a sequential procedure: it operates by maximizing the probability of correctly ranking the top 1 document with a weight assigned to the top position. Subsequently, it focuses on maximizing the probability of correctly ranking the i-th document, considering the corresponding position weight, assuming that the top $i-1$ documents have been ranked correctly. This loss function of the process is formally defined as:

$$\begin{aligned} \mathcal{L}_p(x, \pi_y; h_\psi) = & -\alpha(1) \log P(y^{-1}(1) \mid x; h_\psi) - \\ & \sum_{i=2}^{n} \alpha(i) \log P\left(y^{-1}(i) \mid x, y^{-1}(1), \ldots, y^{-1}(i-1); h_\psi\right), \end{aligned} \tag{12}$$

where $\alpha(\cdot)$ is a decreasing function, i.e., $\alpha(i) > \alpha(i+1)$.

To ensure consistency with the target metric, such as normalized discounted cumulative gain (NDCG), $\alpha(\cdot)$ is defined as the gain function $\alpha(i) = Gain(i) = 2^{n-i} - 1$, which assigns larger weights to documents with higher relevance grades. Combining the Plackett-Luce model (7) with the above loss (12), the optimization objective is

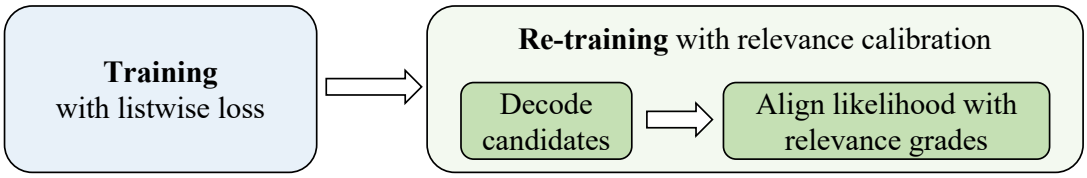

Fig. 2. Overview of the two-stage listwise learning methods, which consists of a training stage using listwise loss and a re-training stage with relevance calibration based on the trained model.

to minimize the following likelihood loss function:

$$\mathcal{L}_p(x, \pi_y; h_\psi) = \sum_{i=1}^{n} \alpha(i) \left(-h_\psi(x_{y^{-1}(i)}) + \log \left(\sum_{k=i}^{n} \exp(h_\psi(x_{y^{-1}(k)})) \right) \right). \tag{13}$$

3 OUR APPROACH

In this section, we present novel listwise generative retrieval models via a sequential learning process. We first provide an overview of our method and then describe the training and re-training stages in detail.

3.1 Overview

In this paper, we propose a *listwise GR approach* (ListGR for short), in which docid lists instead of individual docids are used as instances in learning. ListGR includes a two-stage optimization process, i.e., training with position-aware ListMLE and re-training with relevance calibration. The overall optimization process is illustrated in Figure 2.

To accurately represent listwise relevance, we first establish the position-aware conditional probability of a docid ranked at a particular position with respect to a given query, and employ position-aware ListMLE [48] to train the GR model. To address the decoding inconsistency between the proposed listwise loss function and the beam search decoding, we further retrain the model with relevance calibration techniques for a generated docid list.

3.2 Training with listwise loss function

Inspired by listwise LTR algorithms [11, 48, 91], our key idea is to view the docid ranking problem as a sequential learning process, with each step targeting to maximize the corresponding stepwise probability distribution. In the following, we firstly define the conditional probability distribution of each ground-truth docid with each step, and then apply it to model the ranking list.

3.2.1 ***Positional conditional probability.*** Given a query and its ground-truth docid list π_q, for each docid $id^{(i)}$ in the list, we first obtain the estimated log-probability of generating $id^{(i)}$ for the given query (based on Eq. (4)), regardless of the position of $id^{(i)}$, and perform length normalization, denoted as,

$$\tilde{P}(id^{(i)} \mid q; g_\theta) = \frac{\log \prod_{t \in [1, |id^{(i)}|]} P(w_t \mid q, w_{<t}; g_\theta)}{|id^{(i)}|}, \tag{14}$$

where w_t is the i-th ground-truth token in the $id^{(i)}$, $w_{<t}$ represents the tokens before the i-th one in the $id^{(i)}$. Based on Eq. (14), we further define the positional likelihood, that is, the probability of generation of $id^{(i)}$ being ranked at position i. More specifically, it is the conditional probability distribution of the GR model generating the ground-truth docids from the i-th to the n-th, conditioned on the query. It also represents that the preceding $i-1$

docids are generated at the right positions. The sequential learning process for docid ranking can be summarized as follows:

Step 1: Maximizing the following top-1 positional conditional probability:

$$P(id^{(1)} \mid q; g_\theta) = \frac{\exp(\tilde{P}(id^{(1)} \mid q; g_\theta))}{\sum_{j=1}^{n} \exp(\tilde{P}(id^{(j)} \mid q; g_\theta))}. \tag{15}$$

Please note that Eq. (14) only considers the generation of $id^{(i)}$ conditioned on the query without considering its ranking position in the list , while Eq. (15) requires $id^{(i)}$ to be ranked at the i-th position.

Step i: For $i = 2, \ldots, n$, we maximize the following i-th positional conditional probability,

$$P(id^{(i)} \mid q, id^{(1)}, \ldots, id^{(i-1)}; g_\theta) = \frac{\exp(\tilde{P}(id^{(i)} \mid q; g_\theta))}{\sum_{j=i}^{n} \exp(\tilde{P}(id^{(j)} \mid q; g_\theta))}. \tag{16}$$

The learning process ends at step $n + 1$.

3.2.2 ***Listwise probability with position importance.*** To transform the above sequential optimization problem into a single-objective optimization problem, we define the likelihood of the ground-truth docid list π_q for a query. The likelihood of generating π_q is defined as the product of positional conditional probabilities of different docids. Higher positions are more important and, therefore, we assign the corresponding positional conditional probabilities with higher weights. Therefore, for a query q, the optimization problem is to minimize the probability of generating π_q with negative log likelihood as follows:

$$\begin{aligned} &\min_{g_\theta} -\log P(\pi_q \mid q; g_\theta) \\ &\quad = -\alpha(1) \log P(id^{(1)} \mid q; g_\theta) - \sum_{i=2}^{n} \alpha(i) \log P\left(id^{(i)} \mid q, id^{(1)}, \ldots, id^{(i-1)}; g_\theta\right), \end{aligned} \tag{17}$$

where the weight $\alpha(\cdot)$ is a decreasing function; following [48], we set $\alpha(i) = 2^{n-i} - 1$. Incorporating the probability based on Plackett-Luce model as described in Eq. (15) and 16 into the above optimization problem, we obtain the final listwise loss function:

$$\mathcal{L}_{List}(q, \pi_q; g_\theta) = \sum_{i=1}^{n} \alpha(i) \left(-\tilde{P}(id^{(i)} \mid q; g_\theta) + \log \left(\sum_{k=i}^{n} \exp(\tilde{P}(id^{(k)} \mid q; g_\theta)) \right) \right). \tag{18}$$

The total loss function of a query set Q is $\mathcal{L}_{List}(Q, \pi_Q; g_\theta) = \sum_{q \in Q} \mathcal{L}_{List}(q, \pi_q; g_\theta)$.

Discussions. For retrieval tasks, our listwise approach $\mathcal{L}_{List}$ is different from the pointwise approach used in existing GR works. (i) The existing pointwise approach aims to maximize the likelihood probability of generating relevant docids with MLE for a query. Simultaneously, it suppresses the probability of other irrelevant tokens. When dealing with multi-graded relevance datasets, this method treats docids of different relevance grades as equally important. (ii) In contrast, our listwise approach, $\mathcal{L}_{List}$ loss (Eq. (18)) maximizes the likelihood probability of the ground-truth docid list. Additionally, it assigns corresponding positional weights to different docids based on their relevance grades using p-ListMLE. This enables the model to have better discriminative ability for fine-grained relevance grades. Therefore, our approach aligns more closely with the goal of GR, which is to generate a relevant docid list given a query.

3.2.3 ***Training loss.*** To model the aforementioned listwise relevance, the GR model also needs to learn the fundamental indexing task with the loss defined in Eq. (1) and the retrieval task with the loss defined in Eq. (3). Taken together, the total loss for the training stage is defined as:

$$\mathcal{L}_{Training}(Q, D, I_D, \pi_Q; g_\theta) = \mathcal{L}_{List}(Q, \pi_Q; g_\theta) + \mathcal{L}_{Indexing}(D, I_D; g_\theta) + \mathcal{L}_{Retrieval}(Q, \pi_Q; g_\theta). \tag{19}$$

3.3 Re-training with relevance calibration

After training with a listwise loss function, the GR model gains a better discriminative ability for ranked lists of docids than the previous pointwise approach. However, a decoding inconsistency problem arises [8]. During training, the proposed listwise loss leverages the preceding ground-truth tokens to generate the subsequent token. During inference, the model relies solely on the preceding generated tokens without access to ground-truth tokens. This decoding inconsistency may result in the generated list not being ideal in terms of its ranking according to relevance. Besides, larger beam sizes would cause shorter lengths and worse generation quality [95, 103].

To further improve the quality of the ranked list, we propose to calibrate the generated list, in which the key idea is to align candidates' likelihoods according to their relevance grades to the query. Specifically, we utilize the model trained with Eq. (19) for re-retraining, denoted as $\widehat{g}_\theta$. And for a given query $q \in Q$, a ranked docid list is generated with the beam search strategy, denoted as $\widehat{\pi}_q = [\widehat{id}^{(1)}, \ldots, \widehat{id}^{(n)}]$. We perform both token-level calibration and sequence-level relevance calibration as follows.

3.3.1 ***Token-level relevance calibration.*** For correctly predicted docids, tokens within docids with higher relevance grades are assigned with higher likelihood weights. For incorrectly predicted docids, the generation probability of their tokens should approach zero. Formally, we define the token-level relevance calibration loss as,

$$\mathcal{L}_{Token}(Q, \widehat{\pi}_Q; \widehat{g}_\theta) = -\sum_{q \in Q} \sum_{\widehat{id} \in \widehat{\pi}_q} \sum_{w_t \in \widehat{id}} P_{true}(w_t \mid q, w_{<t}) \log P(w_t \mid q, w_{<t}; \widehat{g}_\theta), \tag{20}$$

where w_t is the t-th generated token in $\widehat{id}$, $w_{<t}$ represents tokens before the t-th token, and $\widehat{\pi}_Q$ is the generated docid list for all queries in Q. Moreover, $P_{true}(w_t \mid q, w_{<t})$ is the weight of generating token w_t computed as follows, given two different candidate docids in π_q, $\widehat{id}^{(i)}$ and $\widehat{id}^{(j)}$:

$$\begin{cases} P_{true}(w_t \mid q, w_{<t}) = 1 - \frac{1}{\left(M\left(\widehat{id}^{(i)}\right)+1\right)^2}, & \forall w_t \in \widehat{id}^{(i)}, \text{ if } \widehat{id}^{(i)} \in \pi_q \\ \sum_{\widehat{id}^{(i)} \in \widehat{\pi}_q} P_{true}(w_t \mid q, w_{<t}) = \beta, & \forall w_t \in \widehat{id}^{(i)}, \text{ if } \widehat{id}^{(i)} \notin \pi_q \\ P_{true}(w_t \mid q, w_{<t}) > P_{true}(w_m \mid q, w_{<m}), & \forall w_t \in \widehat{id}^{(i)} \in \widehat{\pi}_q, w_m \in \widehat{id}^{(j)} \in \widehat{\pi}_q, \\ & \text{if } M\left(\widehat{id}^{(i)}\right) > M\left(\widehat{id}^{(j)}\right) \end{cases} \tag{21}$$

where β is a hyperparameter close to zero, and $M\left(\widehat{id}^{(i)}\right)$ is the relevance grade of $\widehat{id}^{(i)}$, defined in Section 2.1.3. Additionally, w_m represents the m-th token of $\widehat{id}^{(j)}$, and $w_{<m}$ represents tokens before the m-th token in $\widehat{id}^{(j)}$. The effect of each condition of this equation is as follows,

(i) For the first condition, if the generated $\widehat{id}^{(i)}$ is in the ground-truth ranking list π_q, we assign a higher weight P_{true} to this docid, to support its generation. Specifically, this weight is a value less than 1, directly proportional to the relevance grade of the ground-truth label. The higher the relevance grade of the docid, the closer this weight is to 1.

(ii) For the second condition, if the predicted $\widehat{id}^{(i)}$ does not belong to the ground truth docid list, we assign a small weight to suppress its generation. Specifically, this weight β is a small value less than 1, approaching 0.

(iii) For the third condition, for any two docids in the candidate docid list, $\widehat{id}^{(i)}$ and $\widehat{id}^{(j)}$, both belonging to the ground truth ranking list, we adjust their relative weights based on their ground-truth relevance grades. If

the relevance grade of $\widehat{id}^{(i)}$ is higher than that of $\widehat{id}^{(j)}$, then the tokens of $\widehat{id}^{(i)}$ should have higher weights (i.e., $P_{true}(w_t \mid q, w_{<t})$) compared to weights (i.e., $P_{true}(w_m \mid q, w_{<m})$) of $\widehat{id}^{(j)}$.

3.3.2 ***Sequence-level relevance calibration.*** Differences in generation probabilities among distinct docids should correspond to differences in their relevance grades. Docids with higher relevance grades should be prioritized, resulting in a higher likelihood of being ranked higher and generated. Therefore, the sequence-level relevance calibration loss is,

$$\mathcal{L}_{Seq}(Q, \widehat{\pi}_Q; \widehat{g}_\theta) = \sum_i \sum_{j>i} \max\left(0, \widehat{g}_\theta(\widehat{id}^{(j)}) - \widehat{g}_\theta(\widehat{id}^{(i)}) + \lambda_{ij}\right), \tag{22}$$

where $\widehat{g}_\theta(\widehat{id}^{(i)})$ is $P(\widehat{id}^{(i)} \mid q; \widehat{g}_\theta)$ normalized by docid length, i.e.,

$$\widehat{g}_\theta(\widehat{id}^{(i)}) = \frac{\log P(\widehat{id}^{(i)} \mid q; \widehat{g}_\theta)}{|\widehat{id}^{(i)}|^\alpha}, \tag{23}$$

where α is the length penalty hyperparameter, $\forall i, j, 1 < i < j \leq n$, and λ_{ij} is the margin multiplied by the difference in rank position between the docids, i.e., $\lambda_{ij} = (j - i)\lambda$.

3.3.3 ***Re-training loss.*** The final loss of the relevance calibration is defined as:

$$\mathcal{L}_{Re\text{-}training}(Q, \widehat{\pi}_Q; \widehat{g}_\theta) = \mathcal{L}_{Token}(Q, \widehat{\pi}_Q; \widehat{g}_\theta) + \gamma \mathcal{L}_{Seq}(Q, \widehat{\pi}_Q; \widehat{g}_\theta), \tag{24}$$

where γ is the hyperparameter of balancing the two losses.

In summary, our model is first trained using the listwise loss in Eq. (19) and then used to decode ranked docid lists for training queries. After re-training the model using the loss in Eq. (24), inference is performed according to the approach described in Section 2.1.4.

Adaption to binary relevance data. For binary relevance data, since the relevant docids for a query have the same relevance grade, a query may have one or multiple ground-truth docid lists, each containing only one relevant docid, i.e., the top-1 docid. Therefore, in the first training stage, the corresponding position weight $\alpha(1)$ in the listwise loss (Eq. (18)) is set to 0 ($\alpha(i) = 2^{(n-i)} - 1$). Consequently, Eq. (19) reduces into Eq. (5). For binary relevance data, this is acceptable since it only contains docids with the same relevance grade. Our main improvement for the binary relevance data is the relevance calibration Eq. (24) in the re-training stage. It could further optimize the generated candidate docid list, according to docids' relevance grade to the query. In future work, we will explore designing alternative weight functions to enable list-level enhancement for binary relevance data in the first stage as well.

4 EXPERIMENTAL SETTINGS

In this section, we present the experimental settings, including datasets, baselines, model variants, evaluation metrics, and implementation details.

4.1 Datasets

We utilize five widely-used ad-hoc retrieval datasets:

(i) **ClueWeb09-B** (ClueWeb) [18] is a large-scale web collection containing over 50 million documents. The topics are gathered from the TREC Web Tracks conducted from 2009 to 2011.
(ii) **Gov2** [17] consists of approximately 150 queries and 25 million documents collected from the .gov domain web pages. The topics are accumulated from the TREC Terabyte Tracks from 2004 to 2006.

(iii) **Robust04** [86] comprises 250 queries and 0.5 million news articles. The topics of the queries are collected from the TREC 2004 Robust Track.
(iv) **MS MARCO Document Ranking** (MS MARCO) [70] is a comprehensive benchmark dataset for web document retrieval.
(v) **Natural Questions** (NQ) [46] includes natural language questions as queries and Wikipedia articles as documents. Following previous GR studies [10, 84, 87], we perform experiments on the NQ320K version of the dataset, containing 307,000 query-document pairs.

Dataset preprocessing. For multi-graded relevance datasets, i.e., ClueWeb, Gov2, and Robust04 datasets, they are annotated with multi-graded relevance labels, indicating varying degrees of matching with the query intent or information need. Akin to [84], we sample subsets of the original ClueWeb, Gov2, and Robust04 corpora, each of size 200K, for our subsequent experiments. These sampled subsets are referred to as **ClueWeb 200K**, **Gov 200K**, and **Robust 200K**, respectively. The sampling process involves selecting annotated documents first and then randomly choosing additional documents from the remaining corpus, resulting in a total of 200K documents.

For binary relevance datsets, i.e., MS MARCO and NQ datasets, they have documents labeled with binary relevance, indicating whether a document is relevant or irrelevant to a query. For the MS MARCO dataset, following [16, 105], we sample a sub-dataset, **MS MARCO 100K**, consisting of 100K documents, 97K training queries and 3K queries for testing. We sample the training and testing queries from the original training set and development set, respectively. For the NQ320K dataset, following [87], we utilize its open-source preprocessing code.[1] It removes special characters from the documents and performs cleaning and concatenation based on the document structure, such as titles, abstracts, and body text. Table 2 provides statistics of the datasets used in experiments.

Table 2. Data statistics. #Grades denotes the number of relevance grades, e.g., highly relevant and relevant. #Avg denotes the average number of multi-graded relevant documents for queries.

Relevance Type	Dataset	#Queries	#Documents	#Grades	#Avg
Multi-graded	Robust 200K	250	0.2M	2	69
Multi-graded	Gov 200K	150	0.2M	2	180
Multi-graded	ClueWeb 200K	150	0.2M	3	84
Binary	MS MARCO 100K	97K	100K	1	1
Binary	NQ320K	307K	228K	1	1

4.2 Baselines

We first compare our method with traditional retrieval baselines commonly used for document retrieval tasks, including sparse retrieval and dense retrieval methods. The sparse retrieval baselines are:

(i) **BM25** [79] is an effective term-based sparse retrieval method, that represents the classical probabilistic retrieval model.
(ii) **DocT5Query** [73] generates a set of pseudo-queries for each document by a finetuned T5 [77], and then expand the document with these pseudo-queries.
(iii) **SPLADE** [26, 27] uses a BERT to encode the document into a sparse lexical representation.

The dense retrieval baselines are:

(i) **DPR** [41] is a BERT-based dual-encoder model using dense embeddings for text blocks.

[1] https://github.com/solidsea98/Neural-Corpus-Indexer-NCI/blob/main/Data_process/NQ_dataset/NQ_dataset_Process.ipynb

(ii) **ANCE** [93] periodically refreshes the ANN indexer and adpots hard negatives for training a RoBERTa-based dual-encoder model.
(iii) **RepBERT** [99] is a BERT-based two-tower model. And it takes the in-batch negative sampling technique. RepBERT leverages the representation learning capabilities of BERT to represent the query and document, enhancing dense retrieval performance.

Further, we explore several advanced GR methods that are trained in a pointwise manner:

(i) **DSI-Num** [84] uses arbitrary unique numbers as docids. And it uses the MLE loss based on query-docid pairs (Eq. (3)) and document-docid pairs (Eq. (1)).
(ii) **DSI-Sem** [84] generates docids by concatenating category numbers obtained through a hierarchical k-means clustering algorithm. This results in similar documents having similar docids. It shares the same training objective as DSI-Num.
(iii) **DSI-QG** [105] utilizes pairs of pseudo-queries and docids for indexing. The pseudo-queries are generated conditioned on the document using docT5query [73]. Similar to DSI-Num, arbitrary unique numbers are used as docids. DSI-QG can be viewed as DSI-Num with data augmentation techniques.
(iv) **NCI** [87] replaces the arbitrary unique numbers with semantic structured numbers, similar to DSI-Sem. It uses pairs of pseudo-queries and docids, as well as pairs of leading contents of original documents and docids, to train the model. NCI further designs a prefix-aware decoder, which can distinguish the different meanings of the same number in different positions. NCI can be viewed as the DSI-Sem with data augmentation techniques.
(v) **GENRE** [22] retrieves a Wikipedia article by generating its title, specifically designed for the NQ dataset. Due to the absence of titles or incomplete titles in other datasets, we did not experiment with GENRE on those datasets.
(vi) **SEAL** [10] uses arbitrary n-grams in documents as docids and retrieves documents based on an FM-index during inference.

The GR baselines all optimize indexing (Eq. (1)) and retrieval (Eq. (3)) tasks with MLE, so they can all be considered pointwise approaches.

4.3 Model variants

We employ some degraded ListGR models to investigate the effect of our proposed mechanisms:

(i) ListGR$_{pListMLE}$ only trains the model using Eq. (19), and omits the re-training stage.
(ii) ListGR$_{ListMLE}$ replaces the position-wise loss in ListGR$_{pListMLE}$ with the ListMLE loss (Eq. (8)), without considering the position information of docids.
(iii) ListGR$_{Retrain}$ first trains the model using indexing and retrieval loss (Eq. (5)) during the training stage. Then, we perform relevance calibration (Eq. (24)) over the decoded candidate docid lists during the re-training stage.
(iv) ListGR$^{tok}_{pListMLE}$ first trains the model using Eq. (19), and then re-trains the model with the token-level relevance calibration (Eq. (21)).
(v) ListGR$^{seq}_{pListMLE}$ first trains the model using Eq. (19), and then re-trains the model with the sequence-level relevance calibration (Eq. (22)).
(vi) ListGR$_{-aug}$ first trains the model (Eq. (19)) without augmented data, and then perform relevance calibration (Eq. (24)) during the re-training stage.

4.4 Evaluation metrics

For datasets with multi-graded relevance labels, i.e., ClueWeb 200K, Gov 200K, and Robust 200K, we perform 5-fold cross-validation to prevent overfitting while maintaining an adequate number of training instances. The topic titles are used as queries, and the queries are randomly divided into 5 folds. The model parameters are tuned on 4 out of 5 folds, and the remaining fold is used for evaluation. This process is repeated 5 times, with each fold serving as the evaluation set once. The final performance is computed by averaging the results from all tested folds. The evaluation metrics used in this study are normalized discounted cumulative gain (nDCG@K) with $K = \{5, 20\}$, expected reciprocal rank (ERR@20), and precision at rank 20 (P@20), following [13, 32, 66].

For datasets with binary relevance labels, i.e., MS MARCO 100K and NQ320K, we adopt the evaluation metrics used in the original DSI model [84] and subsequent studies [10, 87, 105]. Specifically, we use mean reciprocal rank (MRR@K) with $K = \{3, 20\}$ and hit ratio (Hits@K) with $K = \{1, 10\}$. The performance results are reported on the validation set since the MS MARCO and NQ leaderboards impose restrictions on submission frequency, following [66, 84].

4.5 Implementation details

Model architecture. Following existing GR works [84, 87, 105], we utilize the T5-base model[2] as the backbone for ListGR and the baseline models, for a fair comparison. This particular T5-base model is equipped with a hidden size of 768, a feed-forward layer size of 12, a total of 12 self-attention heads, and a configuration consisting of 12 transformer layers.

Baseline implementation. For BM25, we use the Pyserini [59] implementation for this baseline. For DSI-Num and DSI-Sem, we re-implement these baselines since the source code is unavailable. For other baselines, we use the publicly available source code for experiments.

Docid generation. For the docids used in our work, we leverage semantic structured numbers [84, 87]. Specifically, we apply the hierarchical k-means algorithm introduced in [84] over the document embeddings, which are generated through a 12-layer BERT model with pre-trained parameters, following [84, 87]. First, we cluster all documents into 10 clusters. Then, we recursively apply the clustering algorithm for each cluster that consists of more than 100 documents. The result obtained at each level is used as input for the next level, ensuring a well-organized and manageable clustering process. Finally, for each document, all category numbers obtained at each level are concatenated sequentially as its final docid.

Construction of docid lists. In the five datasets there exist multiple docids at the same relevance grade with respect to a query. During training, we can construct multiple ground-truth docid lists for the query using permutations. The length of the list is determined by the highest annotated relevance grade with respect to the query. Docids within the list are arranged in descending order of relevance grade.

Hyperparameters. Both ListGR and the reproduced baselines are implemented using HuggingFace transformers 4.16.2. For multi-graded relevance datasets, during the training process, we employ the Adam optimizer with a linear warm-up strategy that spans the initial 10% of steps. Our chosen learning rate is set to 6e-5, with a label smoothing factor of 0.01 and a weight decay rate of 0.01. Furthermore, the sequence length of documents is fixed at 512. For binary relevance datasets, the hyperparameter settings are as follows: learning rate is 0.001, batch size is 80, and training steps of 100K. We also adopt Adam optimizer with a linear warm-up strategy that spans the initial 200K steps, label smoothing factor of 0.001, and weight decay rate of 0.02. For all datasets, the maximum

[2]https://huggingface.co/t5-base

number of training steps is capped at 100K, and a batch size of 80 is utilized. To facilitate the training of ListGR, we make use of eight NVIDIA Tesla A100 40GB GPUs, ensuring efficient computation and faster convergence.

Training, re-training and inference. During the training stage, for multi-graded relevance datasets, we set relevance margin λ and docid length penalty α (Eq. (22)) as 0.001 and 0.6, respectively. And during the re-training stage, we set γ used in Eq. (24) to 100, and β used in Eq. (21) to 0.002. For all datasets, to address the limited availability of supervised data, we employ a data augmentation technique that is widely used in existing GR work [16, 76, 82, 83, 87]. Furthermore, following [16, 76, 82, 87], we generate a set of pseudo-queries for all documents to construct additional query-docid pairs for augmentation. Specifically, for MS MARCO 100K, we directly use a publicly trained DocT5query model[3] on the MS MARCO corpus to generate 20 pseudo-queries for each document. For other datasets, we fine-tune a DocT5query model with labled query-document pairs for them to generate 20 pseudo-queries for training, based on the code[4] provided in [105]. DSI-QG, NCI, and our ListGR use same pseudo-queries to enhance the training for a fair comparison. During inference, we construct a decimal trie to constrain the model to decode integers with only 20 beams.

5 EXPERIMENTAL RESULTS

In this section, we report and analyze the experimental results to demonstrate the effectiveness of the proposed ListGR. We target the following research questions:

(RQ1) How does ListGR perform compared with strong retrieval baselines across different relevance scenarios?
(RQ2) How do the training and re-training stages of ListGR affect the retrieval performance?
(RQ3) How does ListGR perform in low-resource settings?
(RQ4) How does the number of relevance grades affect the retrieval performance during training?
(RQ5) How do the model size and beam size affect the efficiency of retrieval?
(RQ6) Can we better understand how different models perform via some case studies?

5.1 Baseline comparison

To answer **RQ1**, we compare ListGR with several representative traditional retrieval methods and some advanced GR methods, in both multi-graded and binary relevance scenarios.

5.1.1 ***Results on multi-graded relevance.*** Table 3 shows the performance of ListGR and baselines on multi-graded relevance datasets. We analyze the results in three parts.

The performance of traditional retrieval baselines. (i) On the three multi-graded datasets, the dense retrieval baseline ANCE outperforms DPR, RepBERT, and sparse retrieval baselines. The reason may be attributed to its ability to learn rich semantic information, and the strategy of using negative samples that aids in acquiring stronger discriminative capabilities than sparse retrieval baselines. (ii) RepBERT exhibits slightly lower performance than BM25 on Gov 200K and Robust 200K, which aligns with findings reported in previous studies [62, 65, 98]. The sub-optimal performance of RepBERT in learning effective query and document representations might be primarily attributed to the limited size of the training set available in Gov 200K and Robust 200K.

The performance of generative retrieval baselines. (i) DSI-Sem surpasses the performance of DSI-Num, while SEAL exhibits even higher performance than DSI-Sem. DSI-Num, DSI-Sem and SEAL use random integers, semantic structured clustering numbers, and n-grams from the documents, respectively. The integration of docids with stronger semantic associations to the document content can significantly enhance the indexing and retrieval effectiveness of GR. This observation aligns with findings reported in previous studies such as [10, 22, 84]. (ii) DSI-QG demonstrates superior performance compared to DSI-Num, DSI-Sem, and SEAL, indicating

[3]https://github.com/castorini/docTTTTTquery
[4]https://github.com/ArvinZhuang/DSI-QG

Table 3. Experimental results on datasets with multi-graded relevance. ∗, †, and ‡ indicate statistically significant improvements over the best performing sparse retrieval baseline SPLADE, dense retrieval baseline ANCE, and generative retrieval baseline NCI, respectively ($p \leq 0.05$).

		nDCG		P	ERR
	Method	@5	@20	@20	@20
ClueWeb 200K	BM25	0.2397	0.2568	0.3221	0.2278
	DocT5query	0.2542	0.2658	0.3363	0.2328
	SPLADE	0.2588	0.2697	0.3371	0.2357
	DPR	0.2672	0.2986	0.3568	0.2806
	ANCE	0.2694	0.3012	0.3587	0.2815
	RepBERT	0.2646	0.2963	0.3520	0.2799
	DSI-Num	0.1520	0.1857	0.2182	0.1167
	DSI-Sem	0.1905	0.2198	0.2563	0.1747
	SEAL	0.2241	0.2355	0.2725	0.1831
	DSI-QG	0.2765	0.2862	0.3604	0.2825
	NCI	0.2885	0.3058	0.3625	0.2863
	ListGR	**0.3341***†‡	**0.3442***†‡	**0.3704***†‡	**0.2928***†‡
Gov 200K	BM25	0.3712	0.3787	0.3379	0.2398
	DocT5query	0.3824	0.3913	0.3435	0.2419
	SPLADE	0.3873	0.3959	0.3486	0.2476
	DPR	0.3864	0.3986	0.3584	0.2496
	ANCE	0.3921	0.4092	0.3605	0.2501
	RepBERT	0.3328	0.3443	0.3076	0.2288
	DSI-Num	0.1525	0.1588	0.1477	0.1360
	DSI-Sem	0.1780	0.1469	0.1516	0.1444
	SEAL	0.2283	0.2053	0.1952	0.1675
	DSI-QG	0.3941	0.4087	0.3635	0.2547
	NCI	0.3986	0.4161	0.3733	0.2629
	ListGR	**0.4153***†‡	**0.4368***†‡	**0.3978***†‡	**0.2824***†‡
Robust 200K	BM25	0.3743	0.3587	0.3456	0.2283
	DocT5query	0.3803	0.3617	0.3549	0.2314
	SPLADE	0.3896	0.3685	0.3573	0.2352
	DPR	0.3917	0.3693	0.3588	0.2371
	ANCE	0.3952	0.3701	0.3592	0.2393
	RepBERT	0.3608	0.3374	0.3244	0.2097
	DSI-Num	0.1649	0.1574	0.1311	0.1205
	DSI-Sem	0.1887	0.1765	0.1508	0.1566
	SEAL	0.2209	0.2093	0.1831	0.1769
	DSI-QG	0.3979	0.3723	0.3615	0.2401
	NCI	0.4012	0.3765	0.3678	0.2435
	ListGR	**0.4284***†‡	**0.3919***†‡	**0.3727***†	**0.2592***†‡

the advantages gained by employing data augmentation techniques that generate additional query-docid pairs. (iii) NCI outperforms DSI-QG due to its use of semantic structured numbers and the presence of the prefix-aware decoder, which effectively distinguishes the meanings of the same numbers in distinct positions within the clustering numerals. (iv) NCI and DSI-QG perform slightly better than ANCE, indicating that using pseudo-queries to enhance learning is crucial for GR models. This has been validated in [76] as well.

The performance of ListGR. By adopting a listwise approach in which lists of docids are used as "instances" in learning, ListGR achieves significantly better performance than existing generative retrieval baselines that work in a pointwise manner. Specifically, on the ClueWeb 200K dataset, ListGR outperforms NCI by 15.8% in terms of nDCG@5. On the Gov 200K dataset, ListGR surpasses NCI by 7.4% in terms of ERR@20. On the Robust 200K dataset, ListGR surpasses NCI by 6.8% in terms of nDCG@5. Furthermore, this outcome suggests that the inclusion of additional relevance levels within the annotated data, such as ClueWeb 200K, yields substantial

Table 4. Experimental results on datasets with binary relevance. ∗, † and ‡ indicate statistically significant improvements over the best performing sparse retrieval baseline SPLADE, dense retrieval baseline ANCE, and generative retrieval baseline NCI, respectively ($p \leq 0.05$).

Methods	MS MARCO 100K				NQ320K			
	MRR		Hits		MRR		Hits	
	@3	@20	@1	@10	@3	@20	@1	@10
BM25	0.3884	0.4157	0.4912	0.5572	0.2849	0.4426	0.2927	0.6015
DocT5query	0.4053	0.4376	0.5029	0.5741	0.3641	0.4825	0.3913	0.6972
SPLADE	0.4164	0.4454	0.5095	0.5813	0.4467	0.7036	0.4982	0.7835
DPR	0.4212	0.4598	0.5214	0.6124	0.4792	0.7583	0.5024	0.8042
ANCE	0.4235	0.4601	0.5327	0.6267	0.4821	0.7622	0.5183	0.8149
RepBERT	0.4202	0.4571	0.5183	0.6052	0.4589	0.7154	0.4835	0.7981
DSI-Num	0.1348	0.1353	0.1264	0.1218	0.1815	0.3785	0.2214	0.4184
DSI-Sem	0.2278	0.2209	0.2123	0.2714	0.2198	0.4248	0.2793	0.5763
GENRE	–	–	–	–	0.3543	0.6218	0.3942	0.7061
SEAL	0.3299	0.3771	0.3721	0.5397	0.3672	0.6398	0.4173	0.7289
DSI-QG	0.4276	0.4524	0.5273	0.6285	0.5834	0.7592	0.6349	0.8236
NCI	0.4359	0.4638	0.5362	0.6396	0.5952	0.7641	0.6425	0.8332
ListGR	**0.4656***†‡	**0.4901***†‡	**0.5576***†‡	**0.6471***†‡	**0.6019***†‡	**0.7723***†‡	**0.6593***†‡	**0.8412***†‡

benefits for ListGR. By incorporating more comprehensive relevance information, ListGR can effectively learn and accurately assess the relevance order among the docid list.

5.1.2 ***Results on binary relevance.*** For the binary relevance datasets, where the positional weight ($\alpha(i) = 2^{n-i} - 1$) of relevant docids is zero, the training stage only utilizes the indexing and retrieval loss (Eq. (5)). Based on this, the trained model undergoes relevance calibration. Table 4 shows the performance of ListGR and baselines on binary relevance datasets. We observe the following: (i) The three dense retrieval baselines outperform sparse retrieval baselines. This could be attributed to the availability of abundant labeled query-document pairs in these two datasets. It helps dense models learn dense representations and captures the semantic relationship between queries and documents. (ii) DSI-Num and DSI-Sem perform worse than dense retrieval baselines, e.g., RepBERT, DPR and ANCE on both binary relevance datasets. This suggests that learning both indexing and retrieval tasks simultaneously through these two types of docids and MLE is still challenging. (iii) SEAL shows better performance than vanilla DSI methods, i.e., DSI-Num and DSI-Sem. The reason might be that SEAL uses n-grams from the documents as docids. This type of docid contains more explicit semantics, which helps the model learn better than numeric docids. (iv) Moreover, both DSI-QG and NCI outperform SEAL, DSI-Num and DSI-Sem, indicating that data augmentation methods, such as transforming documents into pseudo-queries for learning, contribute significantly to the improvement. (v) ListGR outperforms the best-performing GR baseline, NCI, on both binary relevance datasets. Specifically, ListGR achieves improvements of 6.8% in terms of MMR@3 on MS MARCO 100K. This indicates that relevance calibration has the ability to correct inappropriate ordering of docid lists generated by beam search decoding.

5.2 Ablation study

In this section, to answer **RQ2**, we conduct an ablation analysis on three multi-graded relevance datasets to quantitatively assess the impact of each component in ListGR; see Table 5. For the binary relevance datasets, the training stage lacks listwise loss, so that ListGR and ListGR$_{Retrain}$ are the same in this setting; therefore, we did not analyze the performance on binary relevance datasets in this context. We have the following observations:

Listwise loss. (i) ListGR$_{Retrain}$, only using the re-training stage leads to significantly lower performance than ListGR. Additionally, in the training stage, ListGR$_{pListMLE}$ and ListGR$_{ListMLE}$ combining a listwise loss with an

Table 5. Ablation analysis of ListGR with its variants on multi-graded relevance datasets. $*$ indicates statistically significant improvements over all the corresponding variants ($p \leq 0.05$).

Method		nDCG		P	ERR
		@5	@20	@20	@20
ClueWeb 200K	$\text{ListGR}_{pListMLE}$	0.3087	0.3205	0.3618	0.2887
	$\text{ListGR}_{ListMLE}$	0.2947	0.3114	0.3609	0.2874
	$\text{ListGR}_{Retrain}$	0.2961	0.3156	0.3686	0.2881
	$\text{ListGR}^{tok}_{pListMLE}$	0.3224	0.3302	0.3641	0.2894
	$\text{ListGR}^{seq}_{pListMLE}$	0.3252	0.3331	0.3668	0.2908
	ListGR_{-aug}	0.2713	0.2811	0.3509	0.2746
	ListGR	**0.3341**†	**0.3442**†	**0.3704**	**0.2928**
Gov 200K	$\text{ListGR}_{pListMLE}$	0.3998	0.4214	0.3842	0.2765
	$\text{ListGR}_{ListMLE}$	0.3991	0.4185	0.3787	0.2685
	$\text{ListGR}_{Retrain}$	0.3995	0.4192	0.3818	0.2716
	$\text{ListGR}^{tok}_{pListMLE}$	0.4062	0.4256	0.3871	0.2782
	$\text{ListGR}^{seq}_{pListMLE}$	0.4094	0.4288	0.3919	0.2809
	ListGR_{-aug}	0.3551	0.3731	0.3036	0.2204
	ListGR	**0.4153**	**0.4368**†	**0.3978**	**0.2824**
Robust 200K	$\text{ListGR}_{pListMLE}$	0.4074	0.3798	0.3694	0.2483
	$\text{ListGR}_{ListMLE}$	0.4057	0.3778	0.3685	0.2456
	$\text{ListGR}_{Retrain}$	0.4068	0.3783	0.3689	0.2471
	$\text{ListGR}^{tok}_{pListMLE}$	0.4145	0.3826	0.3697	0.2498
	$\text{ListGR}^{seq}_{pListMLE}$	0.4193	0.3851	0.3705	0.2559
	ListGR_{-aug}	0.3528	0.3026	0.2971	0.2066
	ListGR	**0.4284**†	**0.3919**†	**0.3727**	**0.2592**

indexing and retrieval loss improves the retrieval performance over NCI (in Table 3). These results indicate that modeling the ranked docid list explicitly is crucial for better retrieval performance, as using MLE alone does not capture the relationships between docids. (ii) $\text{ListGR}_{pListMLE}$ performs better than $\text{ListGR}_{ListMLE}$, highlighting the importance of position weights in ranking, aligning with the observations in [48]. (iii) ListGR_{-aug} significantly outperforms SEAL (in Table 3). It demonstrates that our listwise approach, even without data augmentation, can assist the GR model in learning stronger discriminative abilitty for relevance.

Relevance calibration. (i) By removing relevance calibration, $\text{ListGR}_{pListMLE}$ and $\text{ListGR}_{ListMLE}$ have a significant drop in performance compared to ListGR. This suggests that beam search decoding has an impact on the inference effectiveness of GR. (ii) Additionally, both $\text{ListGR}^{tok}_{pListMLE}$ and $\text{ListGR}^{seq}_{pListMLE}$, built upon $\text{ListGR}_{pListMLE}$, show improved performance. This indicates that further relevance calibration to candidate docids is essential. (iii) Furthermore, we observe that the performance of $\text{ListGR}^{tok}_{pListMLE}$ and $\text{ListGR}^{seq}_{pListMLE}$ is similar, suggesting that both sequence-level and token-level relevance calibration are crucial for the GR model. These results demonstrate that adjusting the generation probabilities of docids in the candidate docid list generated by the trained model contributes to generating more accurate ranking positions in the list.

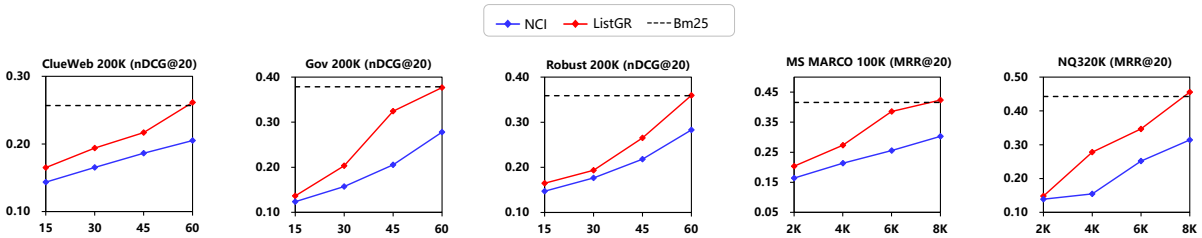

Fig. 3. Training with limited supervision data. The x-axis indicates the number of training queries.

5.3 Low-resource settings

In this section, to answer **RQ3**, during training, we simulate a low-resource retrieval scenario by randomly sampling a fixed and limited number of queries from the training set. More specifically, for the purpose of comparing ListGR and NCI, we randomly sample 15, 30, 45, and 60 queries from the ClueWeb 200K, Gov 200K, and Robust 200K datasets. For the MS MARCO 100K and NQ320K datasets, we randomly sample 2K, 4K, 6K, and 8K queries.

Based on Figure 3, we observe the following: (i) On multi-graded relevance datasets, ListGR outperforms NCI, which suggests that ListGR is capable of modeling the relevance of docid lists using limited information. (ii) Similarly, on binary relevance datasets, ListGR achieves better performance than NCI, indicating that the relevance calibration stage can further enhance the model's ability to recognize the relevance order of docids within the list, even under the pointwise training objective. (iii) ListGR exhibits superior performance compared to a strong BM25 baseline on most datasets. For example, on the ClueWeb 200K dataset, ListGR achieves comparable performance with 58 queries in terms of nDCG@20, while on the MS MARCO 100K dataset, ListGR performs well with only 8% queries, i.e., 7.8K queries in terms of MRR@20.

5.4 Analysis of the relevance grades

To answer **RQ4**, we conduct an analysis by controlling the number of relevance grades employed in the listwise loss during the training phase. This investigation assesses the influence of different numbers of relevance grades on the performance of ListGR.

Specifically, we conduct experiments on the ClueWeb 200K dataset using three, two, and one relevance grades in Eq. (17), respectively. For the case of using two relevance grades, we further divide it into three scenarios: using 2- and 3-grades, using 1- and 3-grades, and using 1- and 2-grades for training. Using only one relevance grade data is equivalent to training with MLE alone (Eq. (5)), which has the same effect as ListGR$_{Retrain}$. During testing, we uniformly use the original testing set consistently across all the aforementioned scenarios.

Based on Figure 4, we observe the following: (i) On the same dataset, increasing the number of relevance grades used in the listwise loss (Eq. (17)) during the training stage leads to better performance. For example, using three relevance grades (blue bar) yields a higher nDCG@20 value than using two (green bars) or one (orange bar) relevance grades only. This could be because providing more relevance labels allows the model to learn more comprehensive and fine-grained differences in relevance. (ii) Among the scenarios using two relevance levels, incorporating 3-graded data results in better performance. For instance, both scenarios using 2- and 3-grades, and using 1- and 3-grades have higher nDCG@20 values than the scenario using 1- and 2-grades. This suggests that docids with higher relevance grades may carry more importance in the list, and learning these docids contributes to better docid list generation.

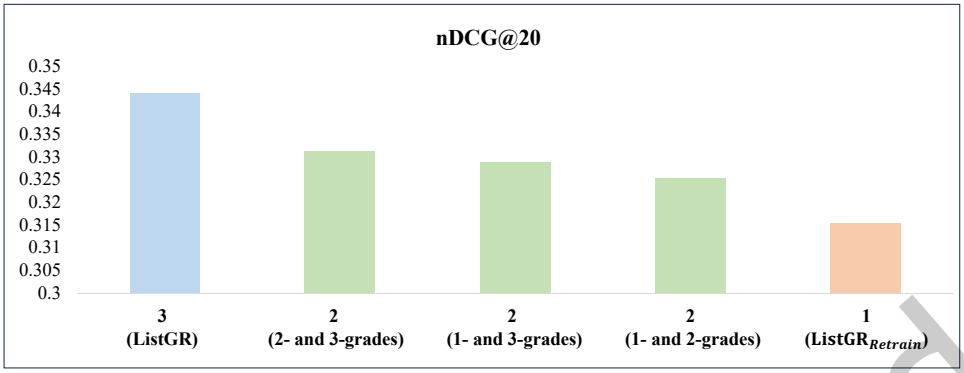

Fig. 4. During the training stage, different numbers of relevance grades in the ClueWeb 200K dataset are used in the listwise loss (Eq. (17)). The x-axis represents the number of relevance grades used, indicated in parentheses as the combination of the relevance grades or the corresponding model names.

5.5 Efficiency analysis

To answer **RQ5**, we analyze the efficiency using an NVIDIA A100-40GB GPU. It is important to note that the inference speed of ListGR is influenced by two factors: model capacity and beam size. In order to provide comprehensive insights, following [87], we have included the latency and throughput measures for various settings in Table 6. Specifically, latency refers to the time it takes for a retrieval model to process a query. And throughput represents the speed at which a retrieval model can process a certain number of queries within a second. Based on the ClueWeb 200K dataset, for latency, we randomly sampled multiple batches of queries, measured the total time for inference, and then divided it by the number of queries to obtain latency. For throughput, we also randomly sampled multiple batches of queries, measured the average number of queries inferenced in 1 second, and obtained the throughput.

Table 6. Efficiency analysis. According to two important factors, namely model size and beam size, ListGR demonstrates encouraging performance in terms of latency and throughput.

Model size	Beam size	Latency (ms)	Throughput (queries/s)
Small	10	76.38	59.73
Base	10	112.56	54.28
Large	10	180.64	45.53
Small	100	218.25	7.81
Base	100	264.07	5.32
Large	100	357.81	4.16

In terms of latency and throughput, ListGR demonstrates promising performance for certain near-real-time applications. The latency of ListGR is comparable to that of DSI [84] when using the same model size and beam size, as both approaches employ beam search with transformer decoders. Similar phenomena is observed in [87]. BM25 has higher retrieval efficiency, but due to a lack of semantic matching, its retrieval performance is lower. RepBERT has lower efficiency because it performs brute-force search based on dense vectors, making it more time-consuming.

Table 7. An example from the ClueWeb 200K dataset, given the query "horse hooves," which has relevant docids with three different grades, ListGR and NCI return the top-5 beams. We also present the corresponding topics and relevance labels of these predicted docids.

	ListGR			NCI		
#Rank	Docid	Topic	Label	Docid	Topic	Label
1	95573	Taking Care Of Horse's Hooves	3	716310	Horse Care Products	1
2	582003	The Barefoot Horse	2	777805	Horse Information	1
3	729007	Steel Horseshoes	2	729007	Steel Horseshoes	2
4	729707	All About Horses	2	729707	All About Horses	2
5	716310	Horse Care Products	1	777711	Pap test	0

5.6 Case study

To answer **RQ6**, we perform case studies from two perspectives. First, we scrutinize the docid lists generated by various methods for a given query. Second, we employ visualization techniques to assess the representations of the query and its candidate documents.

Textual analysis. We take a sample from the test set of ClueWeb 200K and compare the top-5 docid lists predicted by ListGR and NCI. Since both models use semantic structured numbers as docids, we also summarize the topics of the corresponding documents for better understanding and analysis of the differences; see Table 7. Given the query "horse hooves" (QID: 51), docids predicted by ListGR align with their respective relevance labels. However, NCI fails to predict any docids with a relevance level of 3 and struggles to distinguish the relative order of docids with relevance levels 2 and 1. This indicates that the objective of modeling the docid list in ListGR contributes to generating accurate and high-quality docid lists in GR.

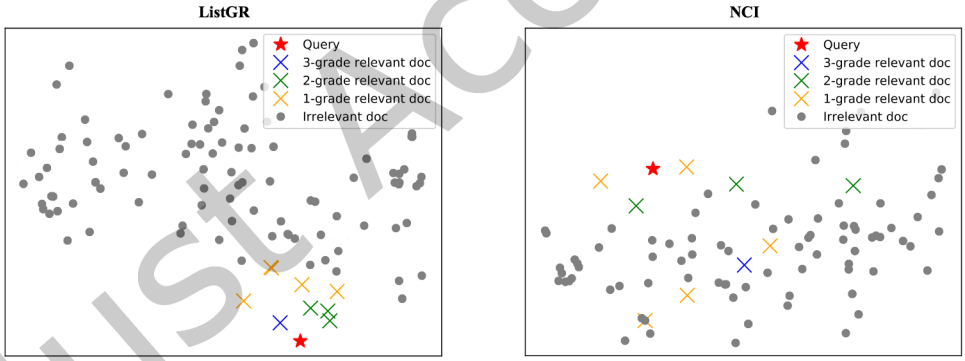

Fig. 5. t-SNE plots of query and document representations for ListGR and NCI. The representations are the output of the encoder of ListGR and NCI.

Visual analysis. To deepen our understanding of ListGR, we employ t-SNE [85] for visualizing the distributions of query and document representations in the semantic space. Expanding on the previous query sample, we create a t-SNE plot to compare the representations of the sampled query and its top-100 candidate documents generated by the encoder output of ListGR and the best-performing GR baseline, NCI [87].

As shown in Figure 5, for ListGR, documents with higher relevance levels are closer to the query, while irrelevant documents are located far away. In the case of NCI, 1-grade relevant documents are closest to the query, while 2- and 3-grade relevant documents are much further away. This demonstrates that ListGR has the ability to differentiate the relevance of docids in a more fine-grained manner in the docid list.

6 RELATED WORK

In this section, we review related work, including the traditional document retrieval, pre-trained language models, and generative retrieval.

6.1 Traditional document retrieval

Document retrieval has traditionally followed an "index-retrieve" paradigm, where documents are indexed and then retrieved based on a query. This paradigm has resulted in two main approaches to document retrieval, namely sparse retrieval and dense retrieval.

6.1.1 ***Sparse retrieval.*** Sparse retrieval methods represent queries and documents using sparse vectors. These methods rely on exact matching to compute similarity scores between queries and documents. In sparse retrieval, the focus is on identifying the presence or absence of specific query terms within documents. Two typical methods in this category are BM25 [79] and the query likelihood model [50]. BM25 takes into account factors such as document length, term frequency, and inverse document frequency to rank documents based on the occurrence of query terms within each document. The query likelihood model [50], on the other hand, leverages a generative model and estimates the probability of generating the query terms given a document. Documents are then ranked based on their likelihood of generating the query. However, these approaches solely consider statistical information and do not incorporate semantic information. To overcome this limitation, several studies [5–7, 20, 28, 104] have utilized word embeddings to reweight the importance of terms. For example, HDCT [21] focuses on long documents. It first utilizes BERT to generate contextual term representations, which are then used to estimate passage-level term weights. Subsequently, these passage-level term weights are aggregated using a weighted sum to obtain document-level term weights. DeepTR [104] constructs a feature vector for query terms and employs a regression model to map these feature vectors to the ground truth weights of terms.

Limitations. Sparse retrieval methods offer computational efficiency due to their reliance on exact matching. They are particularly useful in large-scale retrieval scenarios where the number of documents is substantial. However, these methods often lack the ability to capture semantic relationships and contextual information between query terms and documents, which can limit their retrieval performance.

6.1.2 ***Dense retrieval.*** Unlike sparse retrieval methods that rely on exact matching, which gives rise to the vocabulary mismatch problem [29, 102] dense retrieval focuses on capturing semantic relationships and contextual information [35, 63, 92, 97, 99]. It represents both queries and documents as continuous, dense vectors in a high-dimensional semantic space, to calculate similarity, i.e., using the dot product or cosine similarity as the relevance score.

To enhance the efficiency of dense retrieval, approximate nearest neighbor search methods [4, 9] are employed. These methods accelerate the retrieval process by finding approximate nearest neighbors instead of exact matches. In addition, numerous pre-trained models and techniques have been leveraged to further improve the performance of dense retrieval [1, 12, 33, 42, 53, 71]. For instance, DC-BERT [71] employs dual BERT encoders. In the lower layers, an online BERT encoder is responsible for encoding the query once, while an offline BERT encoder pre-encodes all the documents and stores their term representations in a cache. The obtained contextual term representations are fed into high-layer transformer interaction, initialized by the last few layers of the pre-trained BERT. These approaches take advantage of pre-trained models and advanced techniques to enhance the quality of dense retrieval and can capture more nuanced and subtle semantic relationships between words and phrases in queries and documents, which are often challenging for sparse retrieval methods. To enhance performance, the ranking module is also often leverages [97] . In this work, we focus only on the "index-retrieve" stage,

leaving ranking enhancement for future work. To improve efficiency, approximate nearest neighbor algorithms [30, 39, 93] and various sampling methods [35, 94] have been proposed.

Limitations. Despite the promising performance of the "index-retrieve" paradigm in dense retrieval, there are limitations that need to be addressed: (i) During training, a query encoder and a document encoder are utilized to generate representations for the query and the document, respectively. However, the independence of these encoders restricts the depth of interactions between the representations, thus posing a risk of missing information. Furthermore, the discrete modules in the system cannot be optimized in an end-to-end manner, resulting in sub-optimal performance. (ii) During inference, the query is required to search for relevant documents across the entire corpus. Although efficiency-enhancing strategies are available, such as approximate nearest neighbor search, these methods may sacrifice some semantic information in the process. These limitations highlight the need for further advancements to explore more efficient methods that can retain important semantic information during the retrieval process.

6.2 Pre-trained language models

Pre-trained models have revolutionized natural language processing tasks by leveraging large-scale unsupervised training on vast amounts of text data, with pre-training and fine-tuning techniques [1, 3, 24, 37, 43, 44, 49, 56, 88, 89]. Usually, these models are trained to learn contextualized representations of words, sentences, or documents, which capture rich semantic and syntactic information. Pre-trained models can be broadly classified into two categories, namely discriminative models and generative models.

6.2.1 ***Discriminative pre-trained models.*** Discriminative pre-trained models are primarily designed for tasks that involve classification, regression, or any other form of prediction. Examples of discriminative models include BERT [23], RoBERTa [61], and SpanBERT [40]. Further, they are widely used in IR, for example, BERT is used to re-weight term weights [20, 21, 104] in spare retrieval. Furthermore, dual BERT architectures are used to learn dense query and document representations to support fine-grained semantic interaction [1, 33, 42, 53, 71] in dense retrieval. To bridge the gap between general pre-trained language models and downstream retrieval tasks, some studies [54, 66–68, 92] have proposed specialized pre-training tasks for the retrieval.

6.2.2 ***Generative pre-trained models.*** In addition to discriminative pre-trained models, there has been a growing focus on generative pre-trained models and techniques [2, 36, 47, 60, 77, 100, 101] for text generation. Generative models typically use autoregressive modeling techniques, such as language modeling, where they predict the next word or token in a sequence based on the previous context. Examples of generative models include GPT [74], BART [55], T5 [77]. They have also been researched and applied in IR, for example, T5 is utilized to generate queries for a document. These synthetic queries are then appended to the original documents, creating an "expanded document" to enhance document retrieval [73]. And in [72], given a document, the conditional likelihood of generating queries using GPT serves as the relevance score, which is used for ranking. And dos Santos et al. [25] propose that, given a query and document, T5 concatenates them as input and produces either a "True" or "False" token as output; if the query is relevant to the document, it outputs "True" and proceeds to calculate the generation probability as the relevance score; if the query is irrelevant, it outputs "False".

Limitations. While these explorations with generative models have shown some improvements in information retrieval, some work still revolve around matching queries with documents. This method faces limitations when it comes to dealing with a substantial volume of documents, and it incurs a high computational burden.

6.3 Generative retrieval

In order to further develop the capabilities of generative models, a new retrieval paradigm based on generative models has been proposed, called generative retrieval (GR) [69]. GR aims to directly generate relevant docids for

a qiven query. GR methods parameterize the corpus information, by replacing the traditional external index by a training process that learns the mapping from documents to their corresponding document identifiers (docids). Building upon this framework, researchers have proposed various approaches [10, 14, 15, 22, 52, 57, 78, 82, 84, 87, 96]. GR needs to learn a Seq2Seq model that address two key tasks simultaneously, namely indexing and retrieval.

6.3.1 ***Indexing task.*** In GR this task is aimed at establishing associations between documents and docids. For the document identifiers, in addition to the two primary approaches described in Section 2 – arbitrary unique integers and structured semantic numbers –, there are other types of identifiers. Document titles have garnered considerable attention as they possess inherent semantic relevance [15]. However, methods that use document titles heavily rely on the availability of specific document metadata, limiting their applicability. To address this limitation, some approaches have explored using all n-grams within a passage as its docid [10]. Moreover, the utilization of pseudo-queries generated from the documents as docids has shown significant improvements in retrieval performance [83]. This is because such docids can represent key information about the documents to some extent. Ren et al. [78] leverage tokenized URLs as docids, which may contain key phrases of documents. To provide a more comprehensive representation of the document's information, Li et al. [57] use multiple docids to represent a single document.

To encode the entire corpus, existing approaches primarily employ a Seq2Seq framework, where the original document is taken as input, and the corresponding docid is generated as the output. In this way, the index is embedded within the model parameters, and indexing becomes an integral part of the model training process. Building on [84], we adopt a straightforward input-to-target approach, explicitly associating document tokens with their corresponding docids.

6.3.2 ***Retrieval task.*** In GR this task focuses on mapping queries to relevant docids. Current GR models typically employ a teacher forcing approach [34, 58, 90], maximizing the likelihood of the output sequence conditioned on the input query. If a query has multiple relevant docids, it learns multiple query-docid pairs.

Building upon this blueprint, the first exploration of the GR paradigm was undertaken by GENRE [22]. GENRE utilized the unique titles of Wikipedia articles as document identifiers and employed the BART model [55] to directly generate a list of relevant article titles for a given query using constrained beam search, with a prefix tree of all article titles. This method surpassed some traditional pipelined approaches across various tasks based on Wikipedia. Subsequent research efforts [10, 15, 84, 87, 105] have continued to investigate and enhance the GR paradigm. For example, Zeng et al. [96] design a multi-stage training strategy to generalize GR from moderate-scale datasets [46] to large-scale datasets [70].

Advantages. The GR paradigm offers several advantages: (i) It enables end-to-end optimization, allowing the model to be trained towards the global objective. This means that the entire retrieval process, including both document representation and ranking, can be optimized jointly. (ii) During inference, given a query, the generative model generates docids based on a small-sized vocabulary with beam search. This approach improves retrieval efficiency by eliminating the need for a heavy traditional index, where all documents in the corpus need to be matched against the query for dense retrieval methods.

Limitations. There are several limitations to the GR paradigm. For example, existing work optimizes the model with query-docid pairs by straightforward MLE, which only supports finding the most relevant docids. For queries with multiple relevant docids with multiple relevance grades, the relative order of these relevant docids in the ranked list is randomized, resulting in sub-optimal overall relevance of the ranked list. In this work, we optimize the ranked docid list in a listwise manner and calibrate the generation probabilities of docids within the ranked docid list generated by a beam search-based strategy. To the best of our knowledge, this work is the first attempt to perform listwise optimization in GR.

7 CONCLUSION AND FUTURE WORK

In this paper, to better align with practical retrieval needs of generating a ranked list of results in response to a query, we propose to directly model ranked docid lists in generative retrieval, so that docid lists instead of invididual docids are used as instances in learning. Inspired by position-aware ListMLE in LTR, and considering the characteristics of GR, we maximize the i-th conditional likelihood of a Plackett-Luce model given the top $i-1$ docids. Furthermore, to address the issue of beam search decoding in GR, we design relevance calibration to optimize the order of docids in the list. By conducting comprehensive experiments, we have substantiated that our approach exhibits superior effectiveness compared to existing GR methods.

ListGR has several limitations that give rise to interesting lines of future work:

(i) This work represents our initial exploration of listwise GR, and there are many other listwise approaches [11, 91] in the LTR literature. In the future, we will continue to explore and optimize this work from multiple perspectives. For example, we will investigate how to design position weights in the loss function from a theoretical perspective to make it more suitable for specific use cases. Additionally, we may generate the entire list using a single beam instead of multiple beams, in order to alleviate the impact of beam search decoding on performance.

(ii) In this study, we did not extensively address the design of docids. It is worth noting that the choice of docids can significantly impact both the learning process and retrieval effectiveness. Similar to most existing GR approaches, we assumed that docids are unrelated to the retrieval model and did not optimize them. It is desirable to incorporate the generation and optimization of docids into the model optimization process, allowing for joint learning of docids that are well-suited for GR.

(iii) The paper emphasizes modeling relevance at the list level but acknowledges that relevance should not be the sole focus [81]. LM-based search systems prioritize technology over user-centric aspects, necessitating further development in user interaction and personalization modules. Addressing bias and ensuring controllable and trustworthy search systems are also important topics, along with traceability and interpretability of retrieval structures.

REPRODUCIBILITY

To facilitate reproducibility in this paper, we have only used open datasets. Detailed experimental results and settings are available at https://github.com/lightningtyb/ListGR.

ACKNOWLEDGMENTS

This work was funded by the Strategic Priority Research Program of the CAS under Grants No. XDB0680102, the National Key Research and Development Program of China under Grants No. 2023YFA1011602, the project under Grants No. JCKY2022130C039, the Lenovo-CAS Joint Lab Youth Scientist Project, the CAS Project for Young Scientists in Basic Research under Grant No. YSBR-034, and the Innovation Projectof ICT CAS under Grants No. E261090.

This work was also (partially) funded by the Hybrid Intelligence Center, a 10-year program funded by the Dutch Ministry of Education, Culture and Science through the Netherlands Organisation for Scientific Research, https://hybrid-intelligence-centre.nl, project LESSEN with project number NWA.1389.20.183 of the research program NWA ORC 2020/21, which is (partly) financed by the Dutch Research Council (NWO), and the FINDHR (Fairness and Intersectional Non-Discrimination in Human Recommendation) project that received funding from the European Union's Horizon Europe research and innovation program under grant agreement No 101070212.

We thank our anonymous reviewers for helpful and constructive feedback that helped us to improve the paper. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Samira Abnar, Mostafa Dehghani, Behnam Neyshabur, and Hanie Sedghi. 2021. Exploring the Limits of Large Scale Pre-training. In *International Conference on Learning Representations*.

[2] HeeSun Bae, Seungjae Shin, Byeonghu Na, JoonHo Jang, Kyungwoo Song, and Il-Chul Moon. 2022. From Noisy Prediction to True Label: Noisy Prediction Calibration via Generative Model. In *International Conference on Machine Learning*. 1277–1297.

[3] Mislav Balunovic, Dimitar Dimitrov, Nikola Jovanović, and Martin Vechev. 2022. Lamp: Extracting Text from Gradients with Language Model Priors. *Advances in Neural Information Processing Systems* (2022), 7641–7654.

[4] Jeffrey S. Beis and David G. Lowe. 1997. Shape Indexing Using Approximate Nearest-neighbour Search in High-dimensional Spaces. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition*. 1000–1006.

[5] Michael Bendersky, Donald Metzler, and W. Bruce Croft. 2010. Learning Concept Importance Using a Weighted Dependence Model. In *Proceedings of the third ACM international conference on Web search and data mining*. 31–40.

[6] Michael Bendersky, Donald Metzler, and W. Bruce Croft. 2011. Parameterized Concept Weighting in Verbose Queries. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. 605–614.

[7] Michael Bendersky, Donald Metzler, and W. Bruce Croft. 2012. Effective Query Formulation with Multiple Information Sources. In *Proceedings of the fifth ACM international conference on Web search and data mining*. 443–452.

[8] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. *Advances in neural information processing systems* 28 (2015), 1171–1179.

[9] Jon Louis Bentley. 1975. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM* 18, 9 (1975), 509–517.

[10] Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Wen-tau Yih, Sebastian Riedel, and Fabio Petroni. 2022. Autoregressive Search Engines: Generating Substrings as Document Identifiers. In *Advances in Neural Information Processing Systems*. 31668–31683.

[11] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to Rank: From Pairwise Approach to Listwise Approach. In *Proceedings of the 24th international conference on Machine learning*. 129–136.

[12] Wei-Cheng Chang and Yu. 2020. Pre-training Tasks for Embedding-based Large-scale Retrieval. In *International Conference on Learning Representations*.

[13] Olivier Chapelle, Shihao Ji, Ciya Liao, Emre Velipasaoglu, Larry Lai, and Su-Lin Wu. 2011. Intent-based Diversification of Web Search Results: Metrics and Algorithms. *Information Retrieval* 14 (2011), 572–592.

[14] 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 *The 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1448–1457.

[15] Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Yiqun Liu, Yixing Fan, and Xueqi Cheng. 2022. CorpusBrain: Pre-train a Generative Retrieval Model for Knowledge-Intensive Language Tasks. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 191–200.

[16] Xiaoyang Chen, Yanjiang Liu, Ben He, Le Sun, and Yingfei Sun. 2023. Understanding Differential Search Index for Text Retrieval. In *Findings of the Association for Computational Linguistics*. 10701–10717.

[17] Charles L.A. Clarke, Nick Craswell, and Ian Soboroff. 2004. Overview of the TREC 2004 Terabyte Track. In *TREC 2004*. 74.

[18] Charles L.A. Clarke, Nick Craswell, and Ian Soboroff. 2010. Overview of the TREC 2009 Web Track. In *TREC 2009*. 20–29.

[19] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2019 Deep Learning Track. *arXiv preprint arXiv:2003.07820* (2020).

[20] Zhuyun Dai and Jamie Callan. 2019. Context-aware sentence/passage term importance estimation for first stage retrieval. *arXiv preprint arXiv:1910.10687* (2019).

[21] Zhuyun Dai and Jamie Callan. 2020. Context-aware Document Term Weighting for Ad-hoc Search. In *Proceedings of The Web Conference 2020*. 1897–1907.

[22] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive Entity Retrieval. In *International Conference on Learning Representations*.

[23] 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*. 4171–4186.

[24] Xinshuai Dong, Anh Tuan Luu, Min Lin, Shuicheng Yan, and Hanwang Zhang. 2021. How Should Pre-Trained Language Models Be Fine-Tuned Towards Adversarial Robustness?. In *Advances in Neural Information Processing Systems*. 4356–4369.

[25] Cicero dos Santos, Xiaofei Ma, Ramesh Nallapati, Zhiheng Huang, and Bing Xiang. 2020. Beyond [CLS] through Ranking by Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. 1722–1727.

[26] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE v2: Sparse lexical and expansion model for information retrieval. *arXiv preprint arXiv:2109.10086* (2021).

[27] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2288–2292.
[28] Jibril Frej, Philippe Mulhem, Didier Schwab, and Jean-Pierre Chevallet. 2020. Learning Term Discrimination. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1993–1996.
[29] George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. 1987. The Vocabulary Problem in Human-system Communication. *Commun. ACM* 30 (1987), 964–971. Issue 11.
[30] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. 2013. Optimized product quantization. *IEEE transactions on pattern analysis and machine intelligence* 36, 4 (2013), 744–755.
[31] Anirudh Goyal, Alex M. Lamb, Ying Zhang, Saizheng Zhang, Aaron C. Courville, and Yoshua Bengio. 2016. Professor Forcing: A New Algorithm for Training Recurrent Networks. *Advances in neural information processing systems* 29 (2016), 4601–4609.
[32] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. In *Proceedings of the 25th ACM international on conference on information and knowledge management*. ACM, 55–64.
[33] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval Augmented Language Model Pre-training. In *International conference on machine learning*. 3929–3938.
[34] Yongchang Hao, Yuxin Liu, and Lili Mou. 2022. Teacher Forcing Recovers Reward Functions for Text Generation. In *Advances in Neural Information Processing Systems*, Vol. 35. 12594–12607.
[35] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently teaching an effective dense retriever with balanced topic aware sampling. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 113–122.
[36] Drew A. Hudson and Larry Zitnick. 2021. Generative Adversarial Transformers. In *International Conference on Machine Learning*. 4487–4499.
[37] Gonzalo Jaimovitch-Lopez, David Castellano Falcón, Cesar Ferri, and José Hernández-Orallo. 2021. Think Big, Teach Small: Do Language Models Distil Occam's Razor?. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 1610–1623.
[38] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-based Evaluation of IR Techniques. *ACM Transactions on Information Systems* 20 (2002), 422–446.
[39] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2010. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence* 33, 1 (2010), 117–128.
[40] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving Pre-training by Representing and Predicting Spans. *Transactions of the Association for Computational Linguistics* 8 (2020), 64–77.
[41] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. 6769–6781.
[42] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 39–48.
[43] Tomasz Korbak, Hady Elsahar, Germán Kruszewski, and Marc Dymetman. 2022. Controlling Conditional Language Models without Catastrophic Forgetting. In *International Conference on Machine Learning*. 11499–11528.
[44] Tomasz Korbak, Hady Elsahar, Germán Kruszewski, and Marc Dymetman. 2022. On Reinforcement Learning and Distribution Matching for Fine-Tuning Language Models with no Catastrophic Forgetting. In *Advances in Neural Information Processing Systems*, Vol. 35. 16203–16220.
[45] Jolanta Koszelew and Joanna Karbowska-Chilinska. 2020. Beam Search Algorithm for Anti-Collision Trajectory Planning for Many-to-Many Encounter Situations with Autonomous Surface Vehicles. *Sensors* 20 (2020), 4115. Issue 15.
[46] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, and Ankur Parikh. 2019. Natural Questions: A Benchmark for Question Answering Research. *Transactions oof the Association for Computational Linguistics* 7 (2019), 452–466.
[47] Sylvain Lamprier, Thomas Scialom, Antoine Chaffin, Vincent Claveau, Ewa Kijak, Jacopo Staiano, and Benjamin Piwowarski. 2022. Generative cooperative networks for natural language generation. In *International Conference on Machine Learning*. 11891–11905.
[48] Yanyan Lan, Yadong Zhu, Jiafeng Guo, Shuzi Niu, and Xueqi Cheng. 2014. Position-Aware ListMLE: A Sequential Learning Process for Ranking. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*. 449–458.
[49] Hunter Lang, Monica N. Agrawal, Yoon Kim, and David Sontag. 2022. Co-training Improves Prompt-based Learning for Large Language Models. In *International Conference on Machine Learning*. 11985–12003.
[50] Victor Lavrenko and W. Bruce Croft. 2017. Relevance-based Language Models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, Vol. 51. 260–267.
[51] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep Learning. *Nature* 521, 7553 (2015), 436–444.
[52] Hyunji Lee, Jaeyoung Kim, Hoyeon Chang, Hanseok Oh, Sohee Yang, Vlad Karpukhin, Yi Lu, and Minjoon Seo. 2023. Nonparametric Decoding for Generative Retrieval. *The 61st Annual Meeting of the Association for Computational Linguistics* (2023).

[53] Jinhyuk Lee, Minjoon Seo, Hannaneh Hajishirzi, and Jaewoo Kang. 2020. Contextualized Sparse Representations for Real-Time Open-Domain Question Answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 912–919.
[54] Kenton Lee, Ming-Wei Chang, and Kristian Toutanova. 2019. Latent Retrieval for Weakly Supervised Open Domain Question Answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 6086–6096.
[55] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 7871–7880.
[56] Xuechen Li, Florian Tramer, Percy Liang, and Tatsunori Hashimoto. 2022. Large Language Models Can Be Strong Differentially Private Learners. In *International Conference on Learning Representations*.
[57] Yongqi Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. 2023. Multiview Identifiers Enhanced Generative Retrieval. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 6636–6648.
[58] Guan-Yu Lin and Pu-Jen Cheng. 2022. R-TeaFor: Regularized Teacher-Forcing for Abstractive Summarization. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 6303–6311.
[59] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python Toolkit for Reproducible Information Retrieval Research with Sparse and Dense Representations. In *Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*. 2356–2362.
[60] Wanyu Lin, Hao Lan, and Baochun Li. 2021. Generative Causal Explanations for Graph Neural Networks. In *International Conference on Machine Learning*. 6666–6679.
[61] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692* (2019).
[62] Shuqi Lu, Di He, Chenyan Xiong, Guolin Ke, Waleed Malik, Zhicheng Dou, Paul Bennett, Tie-Yan Liu, and Arnold Overwijk. 2021. Less is More: Pretrain a Strong Siamese Encoder for Dense Text Retrieval Using a Weak Decoder. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 2780–2791.
[63] Yi Luan, Jacob Eisenstein, and Kristina Toutanova. 2021. Sparse, Dense, and Attentional Representations for Text Retrieval. *Transactions of the Association for Computational Linguistics* 9 (2021), 329–345.
[64] R. Duncan Luce. 2012. *Individual Choice Behavior: A Theoretical Analysis*. Courier Corporation.
[65] Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, and Xueqi Cheng. 2022. Pre-Train a Discriminative Text Encoder for Dense Retrieval via Contrastive Span Prediction. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 848–858.
[66] Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, Xiang Ji, and Xueqi Cheng. 2021. PROP: Pre-Training with Representative Words Prediction for Ad-Hoc Retrieval. In *Proceedings of the 14th ACM international conference on web search and data mining*. 283–291.
[67] Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, Yingyan Li, and Xueqi Cheng. 2021. B-PROP: bootstrapped pre-training with representative words prediction for ad-hoc retrieval. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1513–1522.
[68] Zhengyi Ma, Zhicheng Dou, Wei Xu, Xinyu Zhang, Hao Jiang, Zhao Cao, and Ji-Rong Wen. 2021. Pre-Training for Ad-Hoc Retrieval: Hyperlink is Also You Need. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. ACM, 1212–1221.
[69] Donald Metzler, Yi Tay, Dara Bahri, and Marc Najork. 2021. Rethinking Search: Making Domain Experts Out of Dilettantes. In *ACM Special Interest Group on Information Retrieval forum*, Vol. 55. ACM New York, NY, USA, 1–27.
[70] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems*.
[71] Ping Nie, Yuyu Zhang, Xiubo Geng, Arun Ramamurthy, Le Song, and Daxin Jiang. 2020. DC-BERT: Decoupling Question and Document for Efficient Contextual Encoding. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*. ACM, 1829–1832.
[72] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. In *Findings of the Association for Computational Linguistics*. 708–718.
[73] Rodrigo Nogueira and Jimmy Lin. 2019. From doc2query to docTTTTTquery. An MS MARCO Passage Retrieval Task Publication. University of Waterloo.
[74] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, and Mishkin. 2022. Training Language Models to Follow Instructions with Human Feedback. *Advances in Neural Information Processing Systems* 35 (2022), 27730–27744.
[75] Robin L. Plackett. 1975. The Analysis of Permutations. *Journal of the Royal Statistical Society Series C: Applied Statistics* 24, 2 (1975), 193–202.
[76] 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?. In *Gen-IR@SIGIR 2023: The First Workshop on Generative Information Retrieval*.

[77] 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. *The Journal of Machine Learning Research* 21, 1 (2020), 5485–5551.
[78] Ruiyang Ren, Wayne Xin Zhao, Jing Liu, Hua Wu, Ji-Rong Wen, and Haifeng Wang. 2023. TOME: A Two-stage Approach for Model-based Retrieval. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 6102–6114.
[79] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline M. Hancock-Beaulieu, and Mike Gatford. 1995. Okapi at TREC-3. In *Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg*. NIST, 109–126.
[80] Tetsuya Sakai, Daisuke Ishikawa, Noriko Kando, Yohei Seki, Kazuko Kuriyama, and Chin-Yew Lin. 2011. Using Graded-relevance Metrics for Evaluating Community QA Answer Selection. In *Proceedings of the fourth ACM international conference on Web search and data mining*. 187–196.
[81] Chirag Shah and Emily M Bender. 2022. Situating search. In *ACM SIGIR Conference on Human Information Interaction and Retrieval*. 221–232.
[82] Weiwei Sun, Lingyong Yan, Zheng Chen, Shuaiqiang Wang, Haichao Zhu, Pengjie Ren, Zhumin Chen, Dawei Yin, Maarten de Rijke, and Zhaochun Ren. 2023. Learning to Tokenize for Generative Retrieval. In *Advances in Neural Information Processing Systems*.
[83] 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 *29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4904–4913.
[84] Yi Tay, Vinh Q Tran, Mostafa Dehghani, Jianmo Ni, and Dara Bahri. 2022. Transformer Memory as a Differentiable Search Index. In *Advances in Neural Information Processing Systems*, Vol. 35. 21831–21843.
[85] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data Using t-SNE. *Journal of Machine Learning Research* 9, 11 (2008).
[86] Ellen M. Voorhees. 2004. Overview of the TREC 2004 Robust Retrieval Track. In *TREC*. 69–77.
[87] Yujing Wang, Yingyan Hou, Haonan 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 *Advances in Neural Information Processing Systems*, Vol. 35. 25600–25614.
[88] Colin Wei, Sang Michael Xie, and Tengyu Ma. 2021. Why Do Pretrained Language Models Help in Downstream Tasks? An Analysis of Head and Prompt Tuning. In *Advances in Neural Information Processing Systems*, Vol. 34. 16158–16170.
[89] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned Language Models are Zero-Shot Learners. In *International Conference on Learning Representations*.
[90] Ronald J. Williams and David Zipser. 1998. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation* 1 (1998), 270–280.
[91] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise Approach to Learning to Rank: Theory and Algorithm. In *Proceedings of the 25th international conference on Machine learning*. 1192–1199.
[92] Shitao Xiao, Zheng Liu, Yingxia Shao, and Zhao Cao. 2022. RetroMAE: Pre-Training Retrieval-oriented Language Models Via Masked Auto-Encoder. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 538–548.
[93] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808* (2020).
[94] Ji Yang, Xinyang Yi, Derek Zhiyuan Cheng, Lichan Hong, Yang Li, Simon Xiaoming Wang, Taibai Xu, and Ed H Chi. 2020. Mixed negative sampling for learning two-tower neural networks in recommendations. In *Companion proceedings of the web conference 2020*. 441–447.
[95] Yilin Yang, Liang Huang, and Mingbo Ma. 2018. Breaking the Beam Search Curse: A Study of (Re-) Scoring Methods and Stopping Criteria for Neural Machine Translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 3054–3059.
[96] Hansi Zeng, Chen Luo, Bowen Jin, Sheikh Muhammad Sarwar, Tianxin Wei, and Hamed Zamani. 2024. Scalable and Effective Generative Information Retrieval. In *The Web Conference*.
[97] Hansi Zeng, Hamed Zamani, and Vishwa Vinay. 2022. Curriculum learning for dense retrieval distillation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1979–1983.
[98] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2020. Optimizing Dense Retrieval Model Training with Hard Negatives. In *The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1503–1512.
[99] 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).
[100] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with Extracted Gap-sentences for Abstractive Summarization. In *International Conference on Machine Learning*. 11328–11339.
[101] Lily Zhang, Mark Goldstein, and Rajesh Ranganath. 2021. Understanding Failures in Out-of-distribution Detection with Deep Generative Models. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*.

12427–12436.

[102] Le Zhao and Jamie Callan. 2010. Term Necessity Prediction. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management*. 259–268.

[103] Yao Zhao, Misha Khalman, Rishabh Joshi, Shashi Narayan, Mohammad Saleh, and Peter J. Liu. 2023. Calibrating Sequence Likelihood Improves Conditional Language Generation. In *The Eleventh International Conference on Learning Representations*.

[104] Guoqing Zheng and Jamie Callan. 2015. Learning to Reweight Terms with Distributed Representations. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. 575–584.

[105] Shengyao Zhuang, Houxing Ren, and Linjun Shou. 2022. Bridging the Gap between Indexing and Retrieval for Differentiable Search Index with Query Generation. *arXiv preprint arXiv:2206.10128* (2022).