Generalizable Generative Retrieval

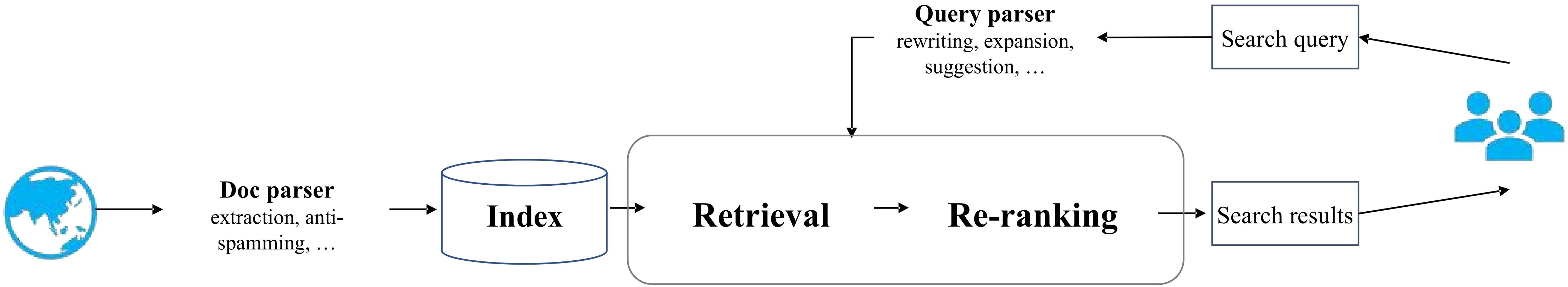
Zhaochun Ren| LIACS

November 14, 2025

Universiteit Leiden
The Netherlands

Pipeline of index-retrieval-ranking

- **Index**: Build an index for each document in the entire corpus
- **Retriever**: Find an initial set of candidate documents for a query
- **Re-ranker**: Determine the relevance degree of each candidate



Closed-book generative retrieval

Passage rank list

Query ---constrained generation---> Document identifiers (doc id)

Query: Who is the singer of *Does He Love You*?

→ **Autoregressive Model** →

Generated: recorded as a duet by American country music artists

Rank 1st passage:
Title: Does He Love You
Body: … recorded as a duet by American country music artists …
ID: 32 – 16 – 18

Rank 2nd passage:
Title: Does He Love You
Body: …
ID: 32 – 16 – 16

Other different designs of document identifiers:

Title: Does He Love You | **ID:** 32 – 16 – 18

[1] Recent advances in generative information retrieval, SIGIR 2024 tutorial

Components of existing GR models

- Document Identifiers design
- Corpus indexing
- Generative decoding from queries

Generative retrieval over dynamic corpora

- Generative Retrieval (GR) shows promise, but its effectiveness in dynamic corpora is largely unexplored.
- Systematically evaluated various current GR models and traditional IR models in dynamic settings.
  - Traditional IR: BM25, DPR, etc.
  - Numeric-based: DSI, GenRET, NCI, etc.
  - Text-based: SEAL, MINDER, etc.

SIGIR 2025
Padova
ITALY

[1] Replication and Exploration of Generative Retrieval over Dynamic Corpora, SIGIR 2025

Experiments

Performance as documents are incrementally added:

Retrieval initial documents.

Method	DocID Type	NQ (Hit@10)						
		$\mathcal{D}_0$	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	$\mathcal{D}_5$	$F_n \downarrow$
Sparse retrieval								
BM25	Term Weight	0.647	0.625	0.611	0.598	0.573	0.573	0.051
Dense retrieval								
DPR	Dense Vector	0.725	0.704	0.696	0.686	0.670	0.660	0.042
DPR-HN	Dense Vector	0.826	0.801	0.797	0.776	0.773	0.768	0.043
Generative retrieval								
DSI-SE	Category Nums	0.718	0.710	0.706	0.702	0.699	0.696	**0.015**
Ultron-PQ	Category Nums	0.795	0.785	0.780	0.780	0.762	0.755	0.023
NCI	Category Nums	**0.871**	0.856	0.844	**0.839**	0.811	0.802	0.041
GenRET	Category Nums	0.858	0.853	0.836	0.829	0.812	0.796	0.033
Ultron-URL	URL Path	0.816	0.810	0.794	0.781	0.780	0.768	0.029
SEAL	N-gram	0.809	0.806	0.788	0.774	0.774	0.763	0.028
MINDER	Multi-text	0.838	0.828	0.813	0.811	0.801	0.773	0.033
LTRGR	Multi-text	0.862	**0.857**	**0.846**	0.827	**0.813**	**0.807**	0.032

Retrieval newly added documents.

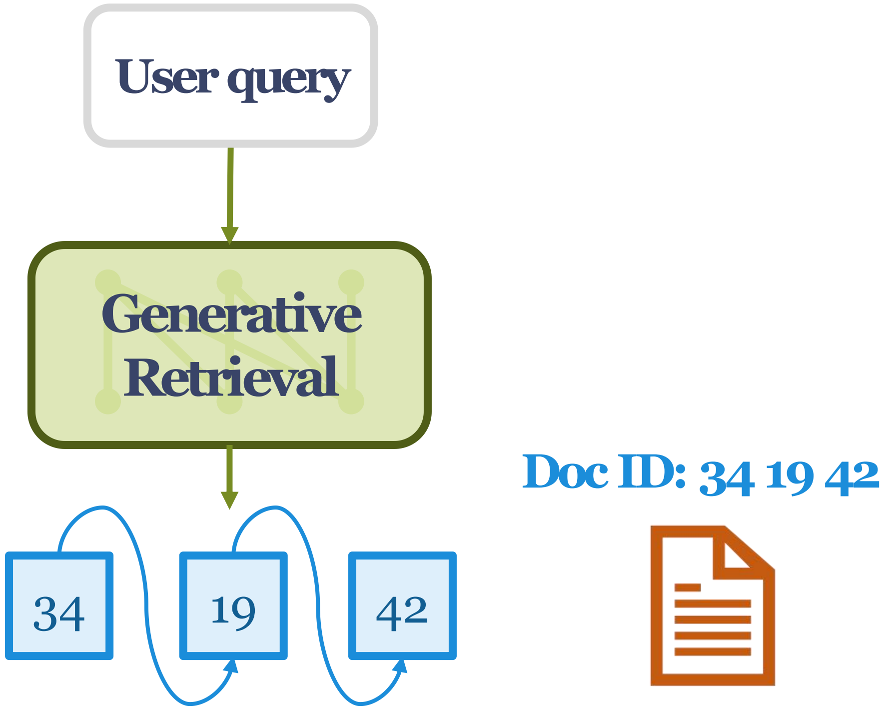
Method	DocID Type	NQ (Hit@10)						
		$\mathcal{D}_0$	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	$\mathcal{D}_5$	$GA_n \uparrow$
Sparse retrieval								
BM25	Term Weight	0.647	0.620	0.588	0.598	0.552	0.571	0.586
Dense retrieval								
DPR	Dense Vector	0.725	0.580	0.587	0.570	0.531	0.544	0.562
DPR-HN	Dense Vector	0.826	0.645	0.644	0.626	0.621	0.624	0.632
Generative retrieval								
DSI-SE	Category Nums	0.718	0.231	0.203	0.221	0.185	0.205	0.209
Ultron-PQ	Category Nums	0.795	0.548	0.549	0.542	0.539	0.532	0.542
NCI	Category Nums	**0.871**	0.464	0.437	0.433	0.358	0.323	0.403
GenRET	Category Nums	0.858	0.361	0.419	0.401	0.357	0.354	0.378
Ultron-URL	URL Path	0.816	0.553	0.545	0.543	0.541	0.532	0.543
SEAL	N-gram	0.809	0.744	0.736	0.727	0.727	0.725	0.732
MINDER	Multi-text	0.838	0.803	0.751	0.746	0.742	0.736	0.756
LTRGR	Multi-text	0.862	**0.831**	**0.803**	**0.811**	**0.779**	**0.773**	**0.799**

- Retrieving Initial Documents:
  - All Methods (BM25, DPR, GR) show stable performance and low forgetting.
  - GR often exhibits good resistance to forgetting, especially numeric-based ones.

- Retrieving Newly Added Documents:
  - BM25 & DPR demonstrate stable generalization ability.
  - GR performance varies Greatly:
    - Numeric-based DocIDs: Poor generalization on new documents (sharp performance drop).
    - Text-based DocIDs (Except Ultron-URL) : Strong generalization on new documents.

Generalization challenge for different corpora

- Related work:
  - Better modeling and training strategies
  - E.g., DSI++, GenRet
- Our work:
  - Impact of constrained generation



Can well-trained GR models generalize directly to different domains?

[1] Constrained Auto-Regressive Decoding Constrains Generative Retrieval, SIGIR 2025

Research problem

- *Can well-trained GR models generalize directly to different domains?*

Awareness of semantics

Can *constrained decoding* handle the domain variation?

We assume the model has perfect knowledge

This is what our work tries to explore

[1] Constrained Auto-Regressive Decoding Constrains Generative Retrieval, SIGIR 2025

Formulation – generative retrieval

- Numerical docID is a *sequence of numbers* from $[k]^m$
- Generative retrieval f is a mapping from user query $\rightarrow [k]^m$
- Let $f(\cdot)$ be the ground truth mapping → Perfect knowledge

[1] Constrained Auto-Regressive Decoding Constrains Generative Retrieval, SIGIR 2025

Formulation – domain-specific corpus

- $[k]^m$ is the ***complete code space*** including all possible docIDs

source

$\downarrow$

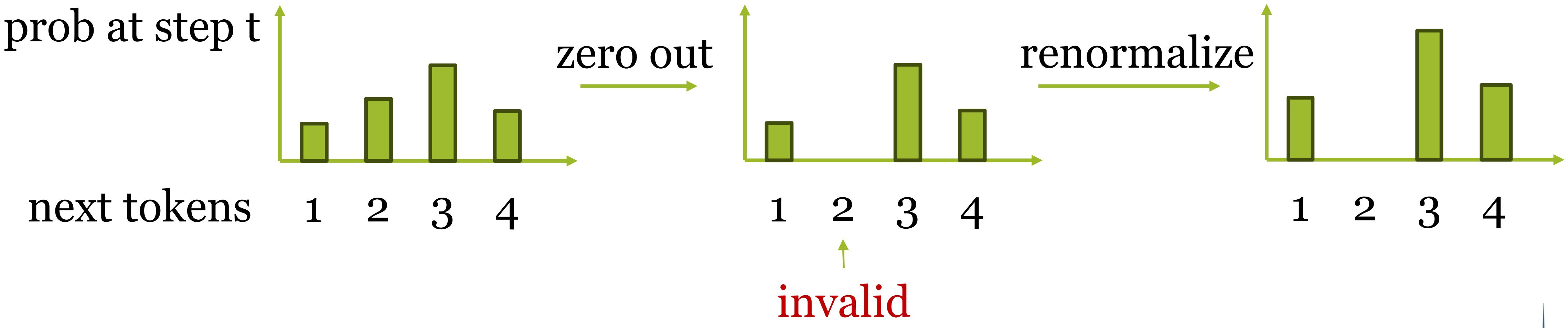
target

- $\mathcal{D} \subseteq [k]^m$ is the ***domain-specific corpus*** that is allowed to generate

[1] Constrained Auto-Regressive Decoding Constrains Generative Retrieval, SIGIR 2025

Formulation – generation

- ***Step-wise constraints*** are used to ensure generating in $\mathbb{D}$

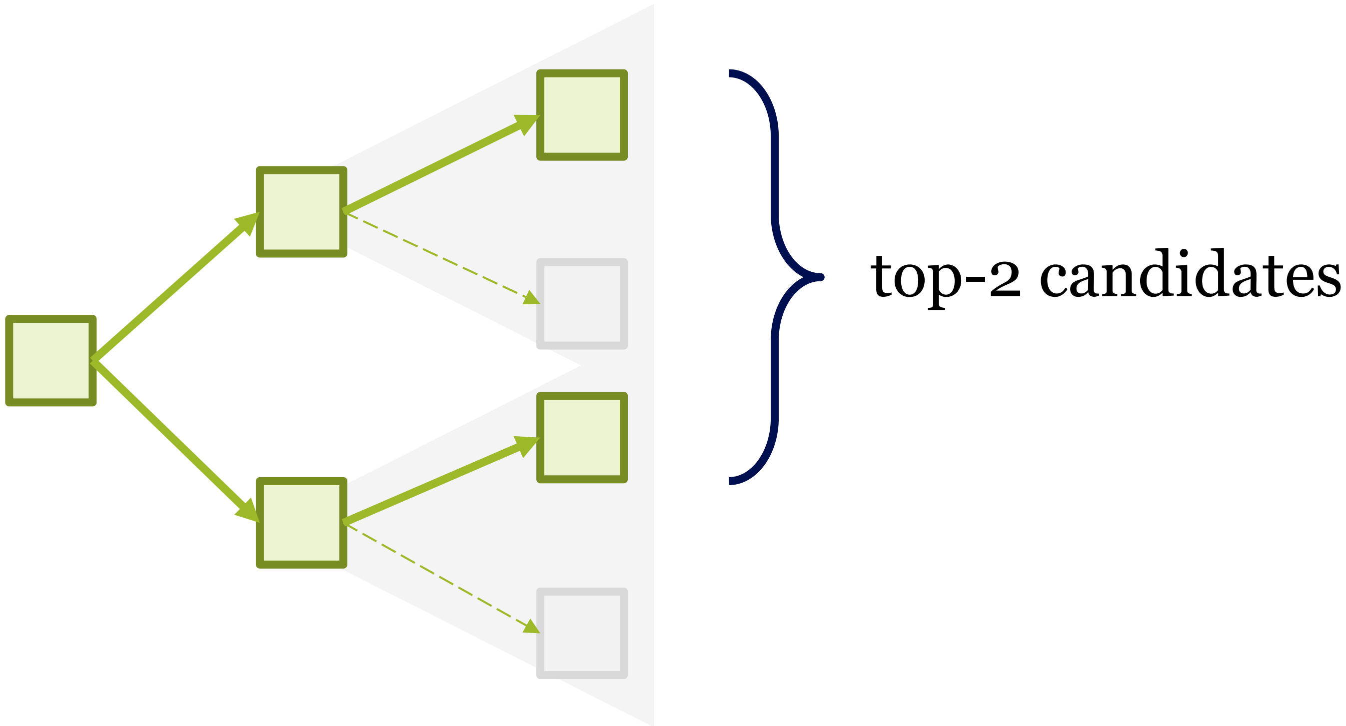


SIGIR 2025
Padova
ITALY

[1] Constrained Auto-Regressive Decoding Constrains Generative Retrieval, SIGIR 2025

Formulation – generation

- ***Beam search*** is used for finding the top-*k* ranked docIDs



[1] Constrained Auto-Regressive Decoding Constrains Generative Retrieval, SIGIR 2025

Our results for research problem

Can a well-trained GR generalize to different domains directly?

RQ1

How does ***step-wise constraint*** affect prediction?

GR has error in prediction of each step

RQ2

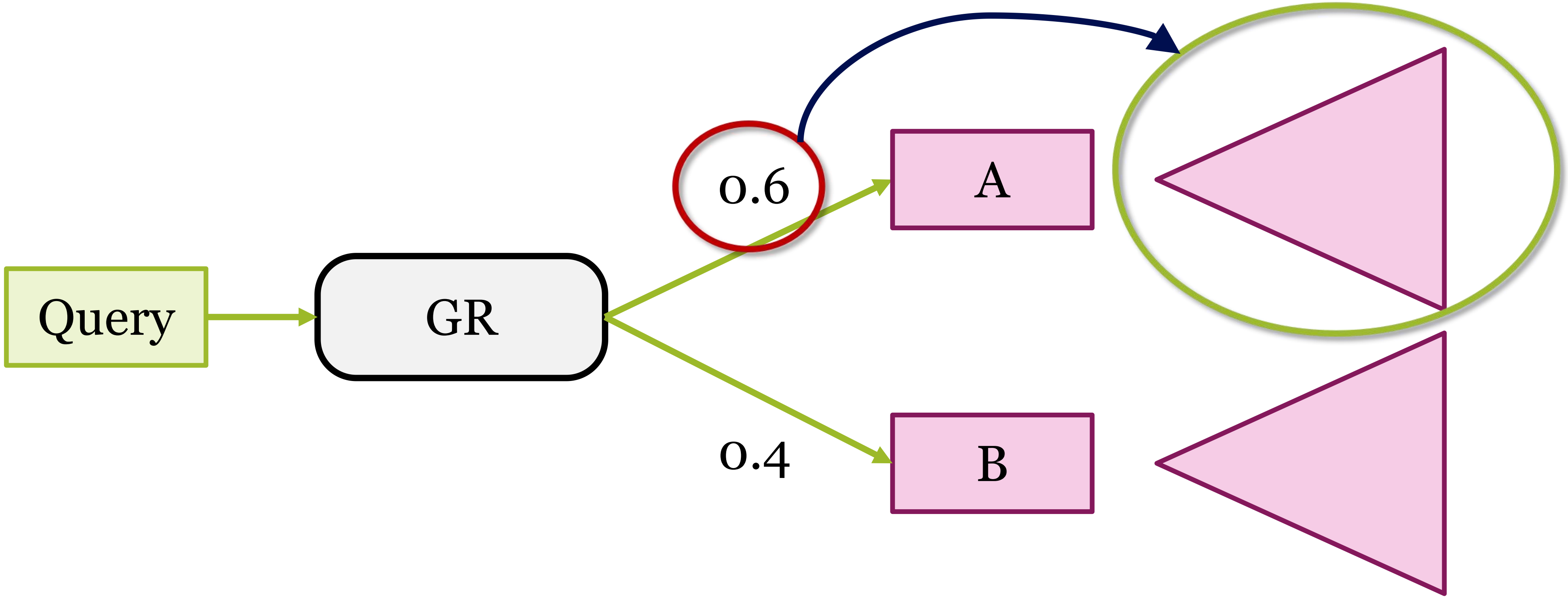
How is ***beam search*** for top-k retrieval?

High precision but low recall

[1] Constrained Auto-Regressive Decoding Constrains Generative Retrieval, SIGIR 2025

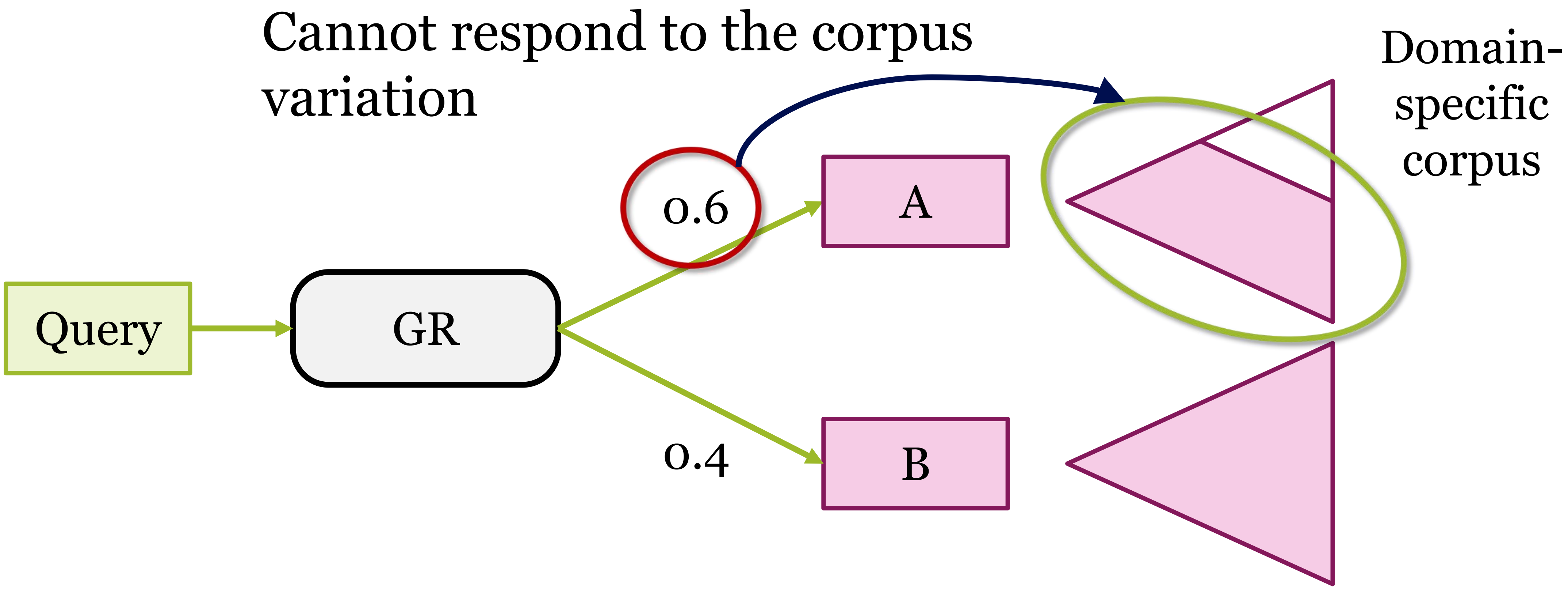
Step-wise constraints – intuition

Model knowledge → Overall estimation of a cluster of docs



[1] Constrained Auto-Regressive Decoding Constrains Generative Retrieval, SIGIR 2025

Step-wise constraints – intuition



[1] Constrained Auto-Regressive Decoding Constrains Generative Retrieval, SIGIR 2025

Step-wise constraints – formal result

Specific case:

uniformly sample domain-specific corpus Ɖ with ratio *p*

Our result:

lower-bound of error computed as KL divergence between model and real next-token distribution

[1] Constrained Auto-Regressive Decoding Constrains Generative Retrieval, SIGIR 2025

Step-wise constraints – formal result

- Formally, we have

$$\mathrm{KL}\left(\boxed{\Pr_c^1} \,\|\, \boxed{f_c^1}\right) \gtrsim \frac{A}{p}$$

A → Degree of concentration

p → Ratio of $\mathfrak{D}$ to $[k]^m$

$\Pr_c^1$ → **Real distribution**

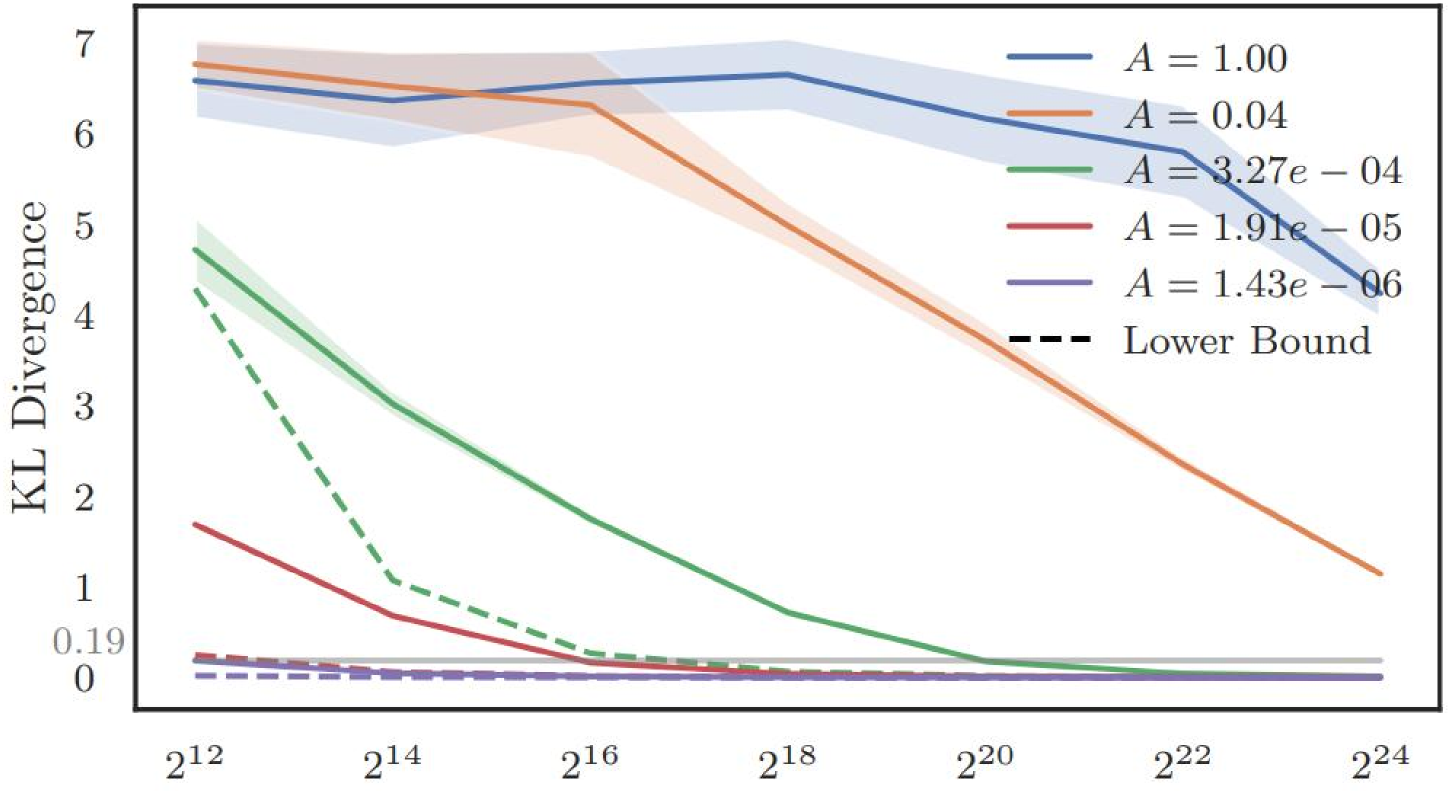
f_c^1 → **Model estimation**

$$A = \mathbb{E}_{d_1}^2\left[\sqrt{\sum_d \Pr(d \mid d_1)^2}\right] \geq \frac{1}{k^{m-1}}$$

Measuring the sharpness of relevance distribution

[1] Constrained Auto-Regressive Decoding Constrains Generative Retrieval, SIGIR 2025

Step-wise constraints – simulation



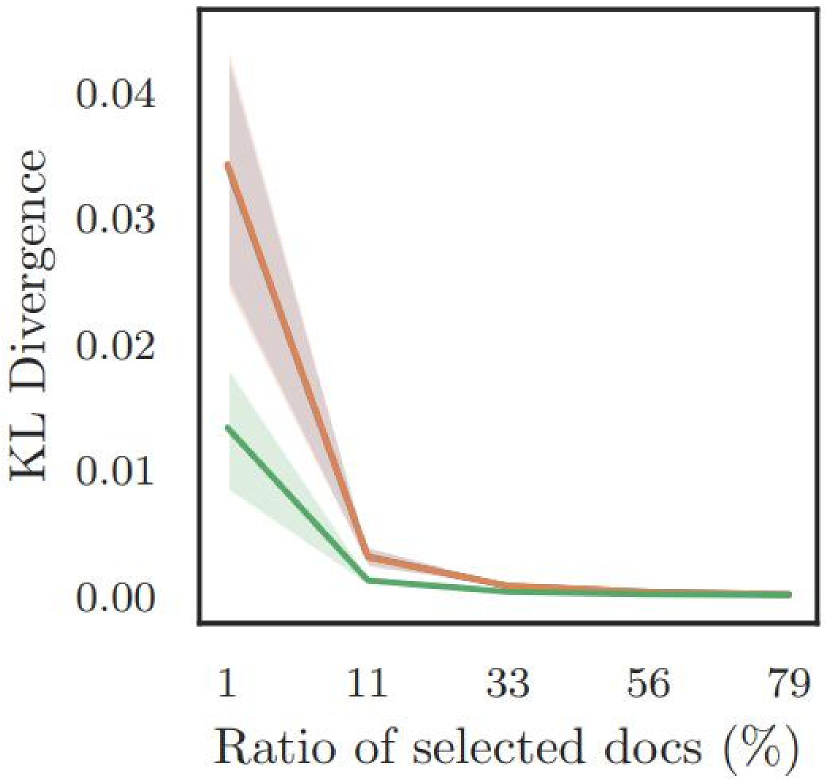
Size of domain-specific corpus

$k = 2^{10}$, $m = 3$

Synthetic distributions with different concentration

Step-wise Constraints – Real World Case

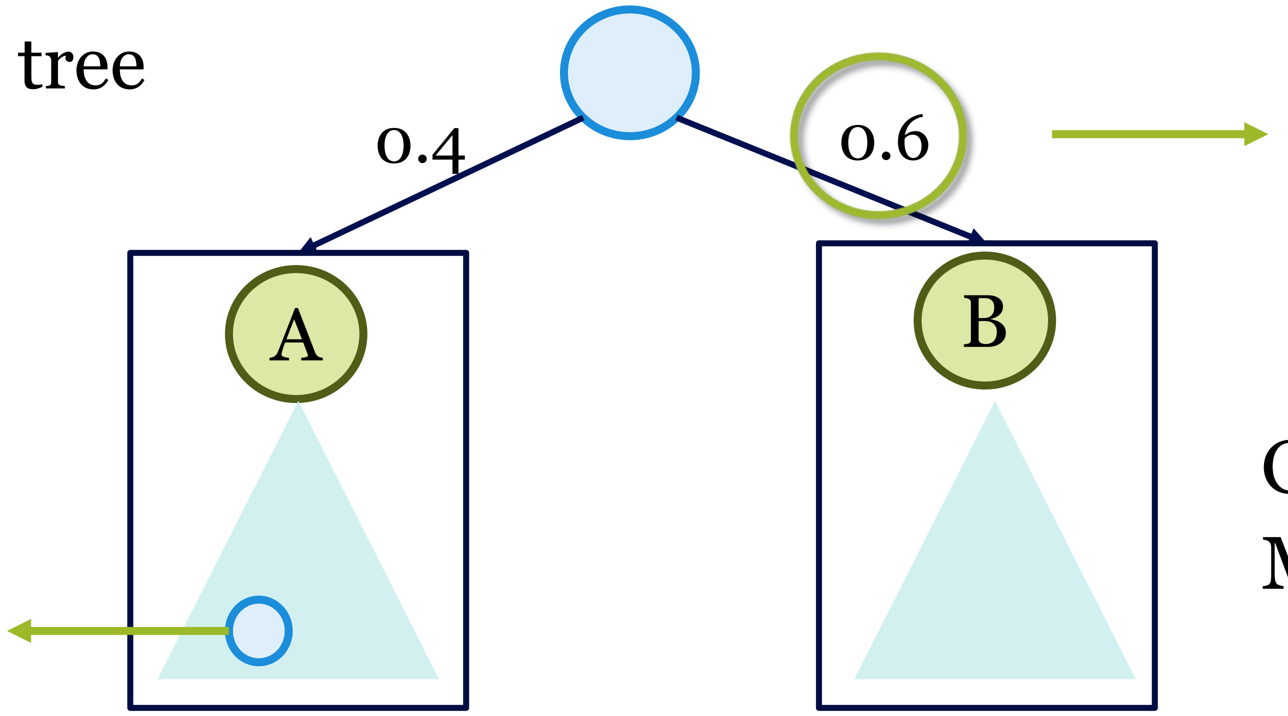
- MS MARCO as the whole corpus
- DocID from Zeng et al. 2024
- Relevance distribution from SLIM++



[1] Constrained Auto-Regressive Decoding Constrains Generative Retrieval, SIGIR 2025

Beam search – intuition

Generation tree



$$\sum_{d \in B} \Pr(d)$$

Relevant doc

Goal: find top-*k*
More suitable: max

Relevant branch   Irrelevant branch

[1] Constrained Auto-Regressive Decoding Constrains Generative Retrieval, SIGIR 2025

Beam search – our result

Relevance distribution:

a. randomly select n docs as relevant

b. relevant score $O(k \log k)$, irrelevant score $o(1)$

Our result (for small n):

the recall of the top-n is upper-bounded by $0.5 + o(1)$ w.h.p.

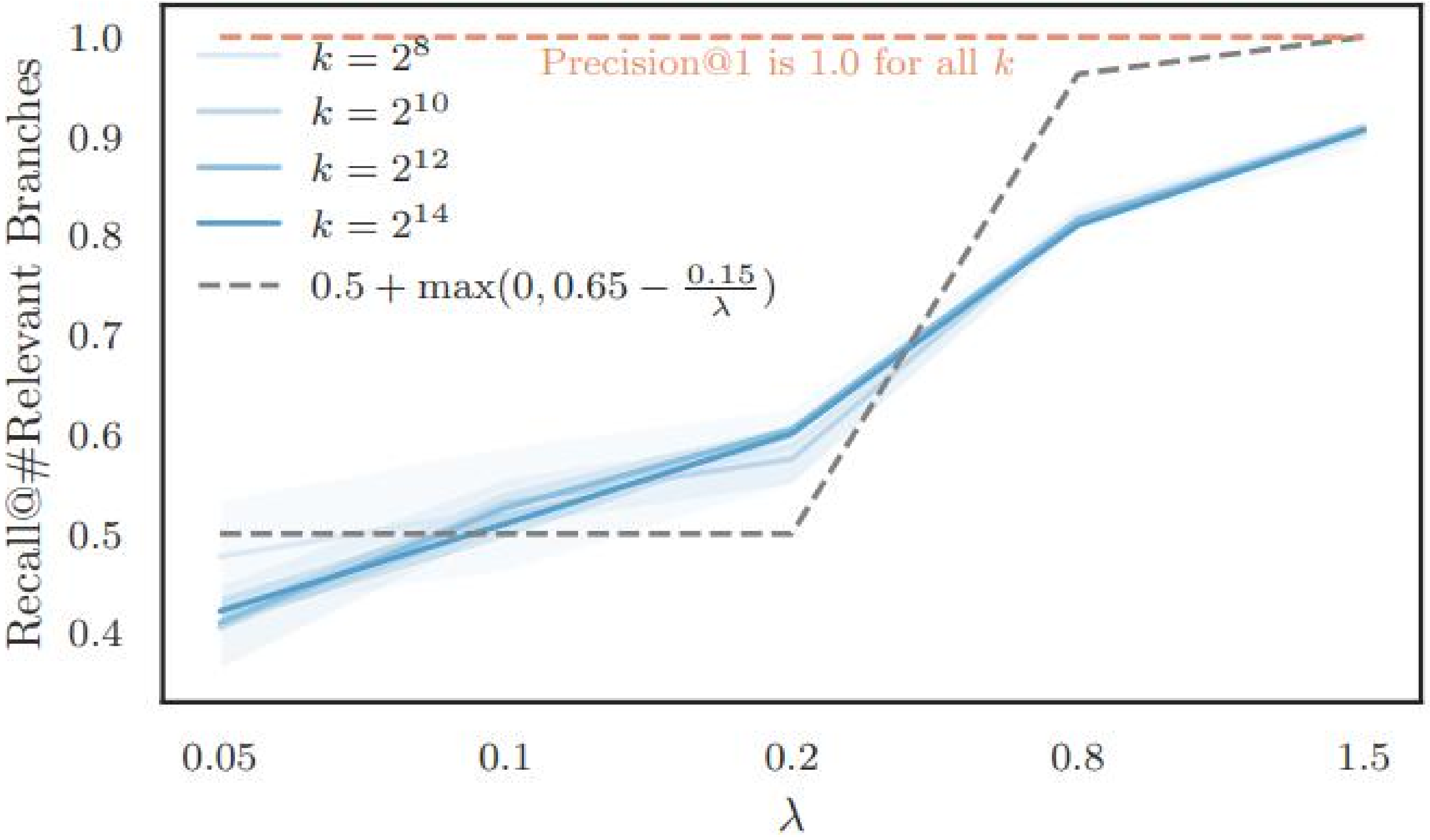
precision@1 is perfect

SIGIR 2025 Padova ITALY

[1] Constrained Auto-Regressive Decoding Constrains Generative Retrieval, SIGIR 2025

Beam Search – Simulation

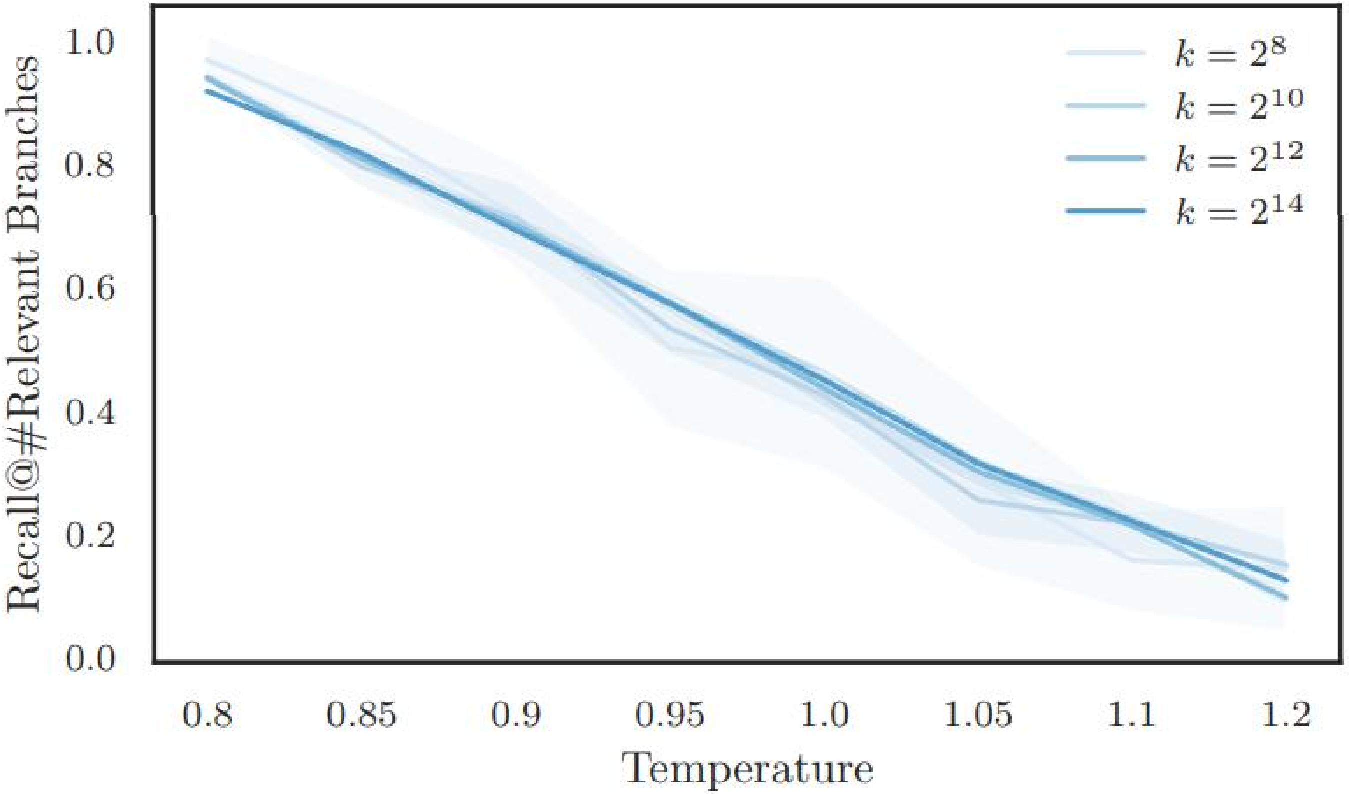
Consider corpus of size k^4, $n = \lambda k$



Number of relevant

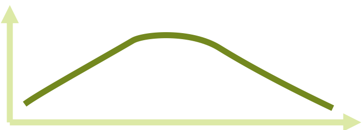
Beam Search – control the sharpness

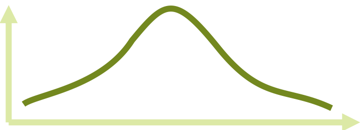
For the score s of a doc, we change it to $s^{\frac{1}{T}}$



[1] Constrained Auto-Regressive Decoding Constrains Generative Retrieval, SIGIR 2025

Trade-off factor – degree of concentration





low → high

Thick tail

Low recall during beam search

Larger variance

High discrepancy in each step

SIGIR 2025 Padova ITALY

[1] Constrained Auto-Regressive Decoding Constrains Generative Retrieval, SIGIR 2025

Implication

A. Generation as retrieval
Model bias exists and can be interfering

B. Likelihood in retrieval
Generative modeling and retrieval task do not fit perfectly somewhere

[1] Constrained Auto-Regressive Decoding Constrains Generative Retrieval, SIGIR 2025

What does constraint GR?

- **Key Challenge – Generalization:**
  - GR models on unseen *out-of-distribution* corpora is not well.
  - The fundamental limitations imposed by GR's **constrained auto-regressive decoding** on generalization remain largely unexplored.
  - Two kinds of generalization challenges:

    Generalization challenge in newly added documents

    Generalization challenge in out-of-distribution IR tasks (unseen tasks)

[1] Constrained Auto-Regressive Decoding Constrains Generative Retrieval, SIGIR 2025

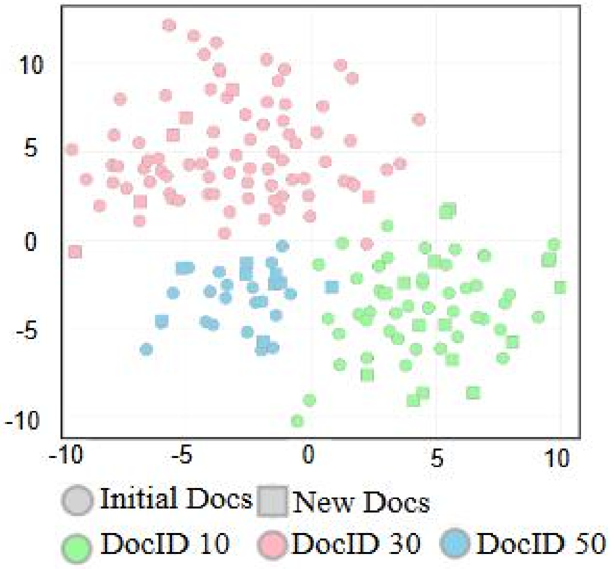
SIGIR 2025
Padova
ITALY

Generalization in newly added documents

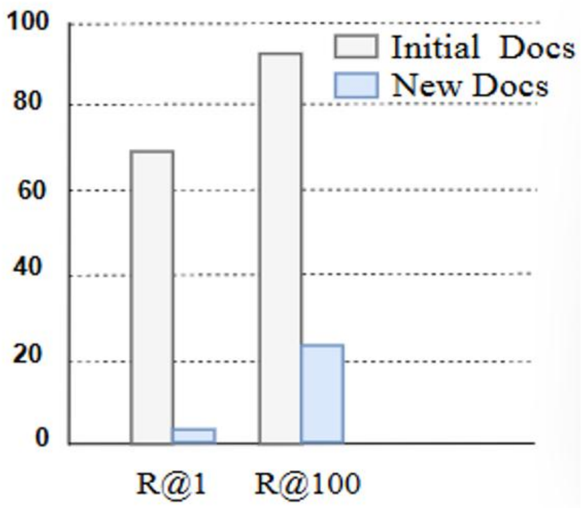
- **New problem framing:** Adapt GR to *new documents* by **editing the semantics→docID mapping**, not full model retraining **(DOME)**.
- **Targeted edit scope:** Use **patching analysis** to locate and edit only parameters responsible for mapping representations to docIDs.
- **GR-aware editing procedure:**
  - **Pseudo-query generation** per new doc to cover diverse intents (many-to-many q–d).
  - **Soft→hard label annealing** to preserve graded relevance patterns while learning unseen docIDs.
- **Forgetting-aware design:** Updates the new mappings **without degrading** existing ones.

DocID mapping bottleneck

- GR models successfully encode semantics of new documents (**a**).
- But fail to generate their correct docIDs (**b**).
- Traditional incremental training is costly and causes catastrophic forgetting.



(a) T-SNE Visualization



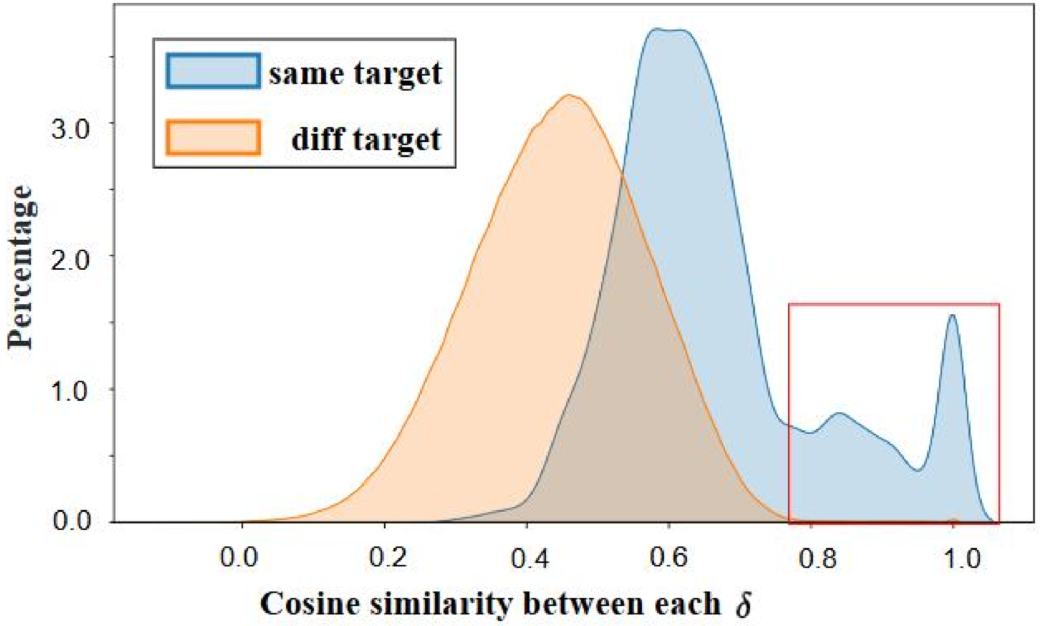
(b) Retrieval Performance

Updata docID mapping parameters

- **Bottleneck**: DocID mapping.
- **Goal**: Selectively updating only the mapping-relevant parameters.
- **Strategy**: Model editing for GR .

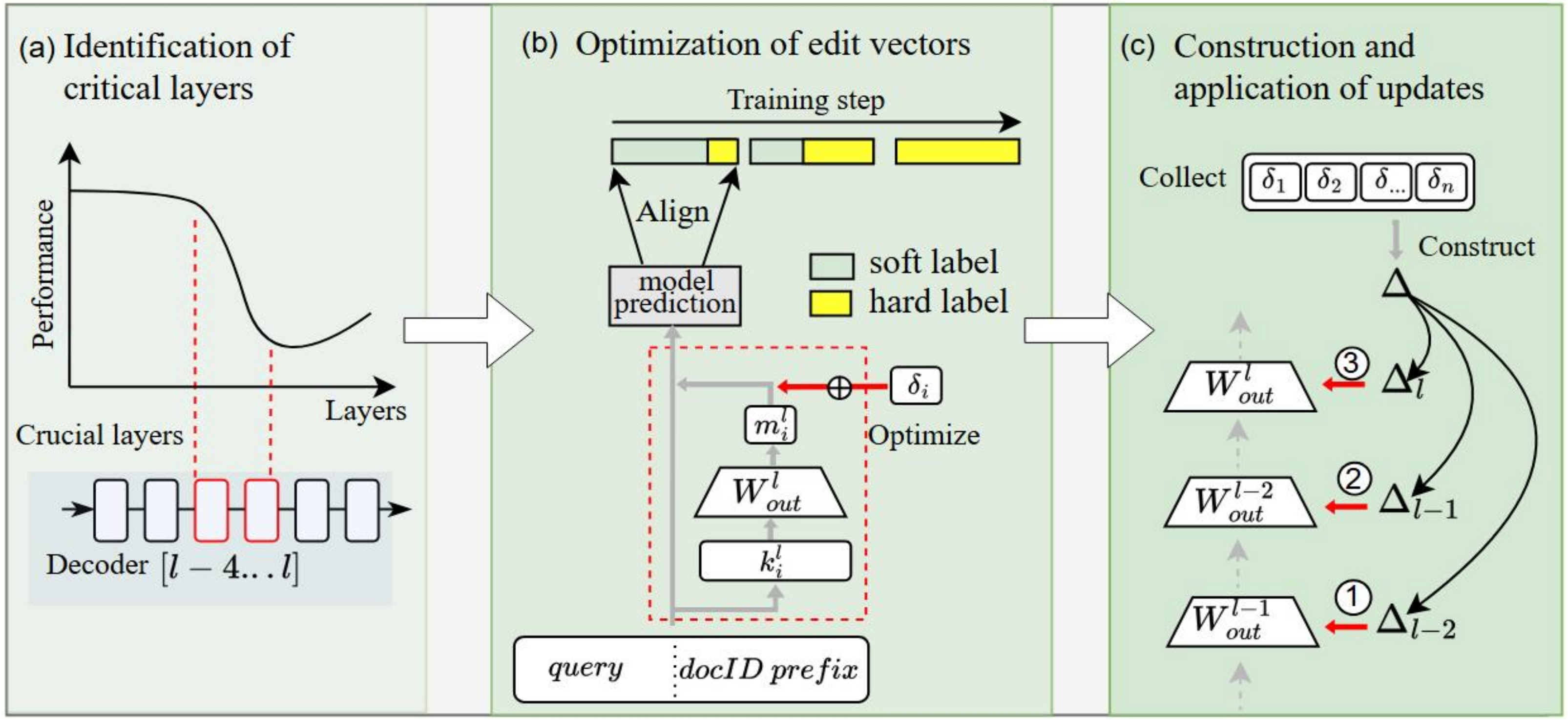
The effectiveness of editing in GR is limited by indistinguishable edit vectors.

Many docIDs share the same target docID
 (e.g., “13-14-17" and "32-16-17" share "-17"),
making their editing vectors indistinguishable.



DOME – DocID-oriented model editing

- (a) Identification of critical layers
- (b) Optimization of edit vectors
- (c) Construct and application of updates

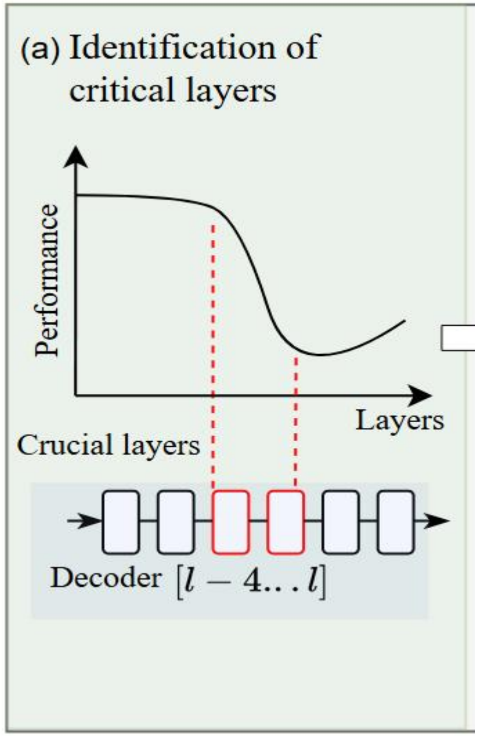


DOME – Identification of critical layers

Pinpoint the decoder layers responsible for DocID mapping.

- Replace a layer's output with its pre-computed average representation.
- Measure the resulting drop in retrieval accuracy.
- Layers causing the largest performance drop are identified as "critical".

All subsequent edits are focused only on these key layers for efficiency and effectiveness.



DOME – Optimization of edit vectors

Generate diverse and effective update vectors δ for new documents.
Challenge: Standard training creates indistinguishable edit vectors for different queries and documents.

Hybrid-Label Adaptive Training

- Create hybrid target label ($\boldsymbol{p_{target}}$)

Blend soft and hard label with an adaptive weight λ:

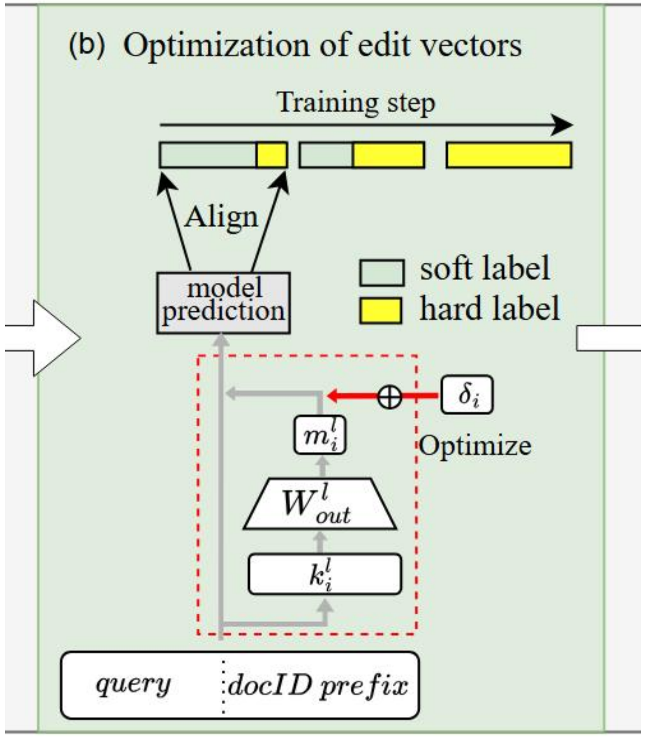
$$p_{target} = (1-\lambda)\underbrace{p_{orig}}_{\text{Soft label (preserves diversity)}} + \lambda \underbrace{\mathbb{1}[v = y_t]}_{\text{Hard label (ensure accuracy)}}$$

λ from 0.3 to 1

- Optimize the edit vector (δ)

$$\delta = \underset{\delta}{\operatorname{argmin}} \left(-\sum_v p_{target}(v) \log p_{edit}(v \mid m_i + \delta)\right)$$

Minimize the cross-entropy loss between the model's new prediction and the hybrid target.



DOME – Construct and application of updates

Precisely inject the new docID mappings into the model's critical layers while preserving its existing retrieval knowledge.

1. Gather key-value pairs

New knowledge (to learn):

- Keys: K_1(FFN inputs from new docs)
- Values: $V_1 = m_1 + \delta$ (FFN outputs with edit vectors)

Old knowledge (to preserve):

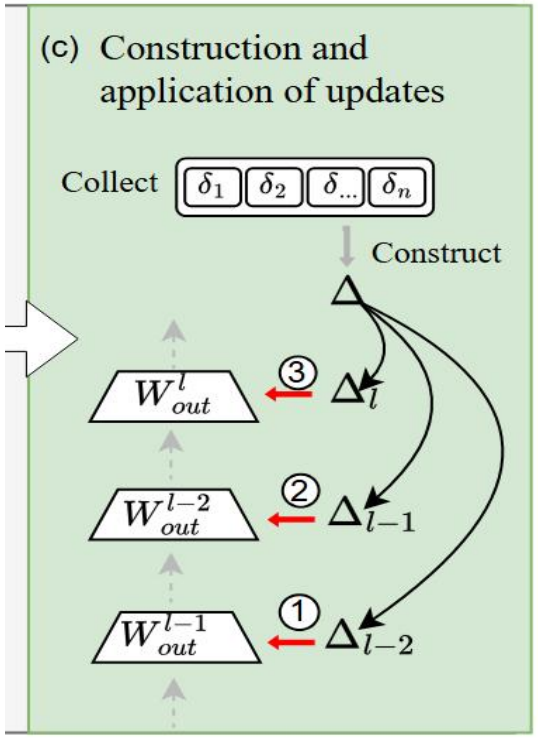
- Keys: K_0(FFN inputs from original docs)
- Values: $V_1 = m_0$(FFN output)

2. Compute update metrix (Δ)

Solve for Δ that maps $K_1 \rightarrow V_1$ while preserving $K_0 \rightarrow V_0$ using a constrained, closed-form solution in model editing.

3. Apply Sequentially Across Layers

Distribute the update based on layer distance: $\Delta_j \propto \frac{1}{dist(j,l)} \Delta$



Main results

New documents.

Method	NQ	MS-MARCO	Time/Doc
	R@1 / R@100	R@10 / MRR@10	(s)
No Training			
DSI	0.07 / 0.25	0.06 / 0.03	-
Incremental			
DSI++	0.67 / 0.92	0.76 / 0.52	3.54
Model Editing			
MEMIT	0.61 / 0.76	0.67 / 0.50	12.62
DOME	**0.69 / 0.93**	**0.76 / 0.52**	**2.14**

Initial documents.

Method	R@1 / R@100	Fn
Base model	0.696 / 0.931	-
Baselines		
New-Doc FT	0.544 / 0.805	0.125
DSI++	0.674 / 0. 926	0.007
Model Editing		
ROME	0.285 / 0.774	0.317
MEMIT	0.647 / 0.904	0.020
DOME	**0.692 / 0.928**	**0.003**

- **Efficiency:** Dramatically **reduces adaptation time** vs. incremental retraining—no full corpus re-indexing or large-scale fine-tuning.
- **Accuracy on new docs:** Significant **Recall@10 / Acc gains** on **NQ** and **MS-MARCO** for newly added items.
- **Robustness: Strong resistance to catastrophic forgetting** on original corpus.

Ablation study

Pseudo Queries

More queries → Richer context → Higher recall. (R@1: .506 → .652)

DocID Assignment

Works with any scheme (RQ, BM25, PQ) → Highly generalizable.

Edited Layer Range

Optimal at layers 14-18 → Confirms our localization is critical.

Hybrid-Label Training

- Hard-Label Only: Lacks diversity (vectors collapse).
- Soft-Label Only: Lacks accuracy (no precise target).
- Hybrid (Both): Achieves diverse and accurate updates.

Variant	R@1	R@10	R@100	MRR@100
DOME (full)	**0.686**	**0.880**	**0.927**	**0.740**
Number of pseudo queries per new doc				
Pseudo = 1	0.506	0.760	0.875	0.592
Pseudo = 4	0.615	0.832	0.898	0.678
Pseudo = 7	0.652	0.866	0.918	0.698
DocID type				
BM25-based docIDs	0.683	0.860	0.921	0.722
PQ-based docIDs	0.675	0.877	0.933	0.719
Edited decoder layer range				
Layers 11–15	0.659	0.857	0.904	0.704
Layers 18–22	0.536	0.717	0.836	0.624
soft-to-hard label strategy				
w/o soft label	0.646	0.805	0.901	0.697
w/o hard label	0.325	0.556	0.711	0.436

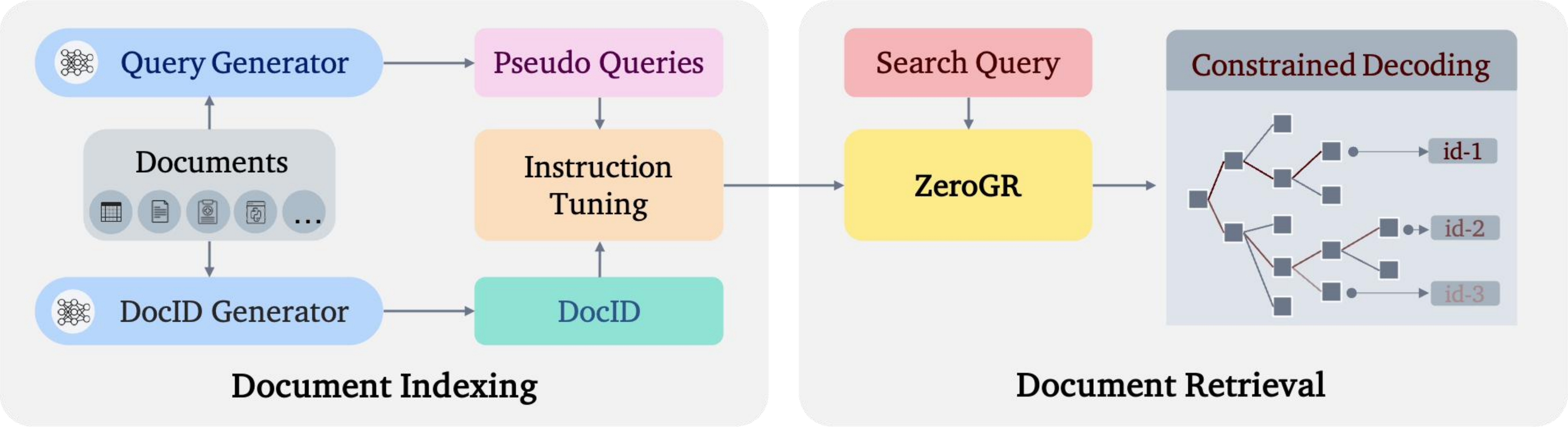
Generalization to unseen tasks

- Generative Retrieval (GR) encodes corpus in parameters → generate docids at query time.
- **Gap:** GR trained in-domain struggles to generalize to *unseen* tasks (zero-shot, heterogeneous corpora, task-specific relevance).

- **Question: How to make GR *generalize* across tasks with no supervision?**

ZeroGR at a glance

- Leverages **natural-language task instructions** to adapt GR without labels.
- Three components:
  - **Unified DocID generator** G_{ψ} → short, keyword-rich docids for any modality (text/tables/code).
  - **Instructed query generator** GF_{θ} → diverse pseudo-queries from task instruction.
  - **Reverse-annealed decoding** → balanced precision/recall when generating ranked docids.



Unified DocID representation

- **Problem:**
  - Titles/URLs/spans don't generalize to custom corpora;
  - RQ-VAE can be unstable.
- **Design:** LM-based generator maps a document to a short (6–8 words) *keyword-centric* sentence ranked by coverage.
- Train data: prompt a powerful LM for ⟨doc, docid⟩ pairs → fine-tune a compact Llama-1B-Instruct.
- **Benefits: Natural language priors, fewer conflicts, faster convergence.**

DocID

Token sequence

Instructed corpus indexing

- **Goal:** Close pseudo-query vs real-query distribution gap in heterogeneous IR.
- Using Instruction-tune Llama-1B on diverse IR tasks verbalized via instructions.
- For each doc d_i & instruction *instr*, sample B pseudo-queries $\{q_{i,1..B}\}$.

Pseudo query

- Train GR index (*LLM*) with cross-entropy on ⟨instr $\oplus$ q, docid⟩.

Reverse-Annealed DocID decoding

- Problem:
  - Beam search collapses diversity → low recall;
  - nucleus sampling improves recall but hurts precision.
- **Reverse annealing:** Sample docids *without replacement* while **increasing temperature** across iterations.
- Constrained by a prefix tree of valid docids; remove leaves after sampling.

- **Effect:** Early high-precision picks, later broader exploration → better overall ranking.

Experimental setups

- Datasets: curated from MAIR training splits + additional instruction-tuning data.
  - **Coverage:** 69 tasks across 6 domains; ~41M query–doc pairs with instructions.
  - **Domain stats (illustrative):** Medical (5), Finance (8), Academic (16), Code (13), Legal (7), Web (17).

MAIR dataset

[1] MAIR: A Massive Benchmark for Evaluating Instructed Retrieval, EMNLP 2024

Experimental setups

- **Evaluation benchmarks:** BEIR (12 tasks) and MAIR (seen vs. **unseen** subsets).
- **Metrics:** Acc@1, nDCG@10, Recall@100.
- **Implementation:**
  - Llama-based models for docid gen, query gen, and GR index;
  - fixed LR 5e-5; 5 epochs for 1B components.
- **Baselines:**
  - **Sparse:** BM25 (BM25S).
  - **Single-task dense:** Contriever-MARCO, GTR-base/large.
  - **Multi-task dense:** E5-Base/Large, BGE-Base/Large, OpenAI-Embed-v3-Small.
  - **Instruction-tuned dense:** E5-Mistral-7B-instr, GritLM-7B.
  - **GR competitors (for BEIR table):** GENRE, GENRET, GLEN, TIGER.

Research questions in the experiments

- **How does ZeroGR compare with state-of-the-art retrieval methods?**
  - We evaluate ZeroGR against leading models on the MAIR benchmark and conduct additional analysis on the BEIR datasets
- **How do model design and training strategies influence the performance of ZeroGR on unseen IR tasks?**
  - a systematic study on the development set, investigating key factors in generative retrieval.

Main results on MAIR and BEIR

- ZeroGR Average **Acc@1 = 41.1** on MAIR
  - above BM25, Contriever/GTR/E5/BGE, and OpenAI-Embed-v3-Small.
- **Unseen subsets:** State-of-the-art on Apple, MB, PM.A, DD, NCL (examples) → robust transfer.
- **Efficiency:** 3B-param GR rivaling/ surpassing 7B instruction-tuned dense retrievers.

- **Average:** ZeroGR **44.9** vs GENRET **41.1**; outperforms on SciFact, FiQA, Covid, etc.
- Per-dataset highlights: best on ArguAna, SciFact, FiQA, Covid; competitive on NFCorpus.

Scaling instruction fine-Tuning

- Increase *task diversity* in instruction-tuning → consistent gains on MAIR-unseen.
- Data ramps: MS MARCO → +OpenQA → +BEIR-Train → +MTEB-Train → +ZeroGR-Train.
- Effects:
  - Longer & more diverse queries (task-aware).
  - Lower docid conflict rate.
  - Higher Acc@1 on unseen tasks.

Scaling query numbers and model size

Takeaways

Takeaways

- **DOME**: docID-oriented model editing that effectively and efficiently adapts GR models to unseen documents.
- **ZeroGR**: instruction-driven GR that generalizes in zero-shot settings across heterogeneous tasks.
- Unified docids + instructed pseudo-queries + reverse-annealed decoding → SOTA GR on BEIR/MAIR.
- Positive scaling trends: task diversity, query count, model size.

Limitations / Future

- multimodal docids;
- decoding theory;
- safety & bias auditing.

Thanks for your attention!

Zhaochun Ren
z.ren@liacs.leidenuniv.nl

Universiteit Leiden
The Netherlands