生成式检索的探索和实践

冯少雄 小红书

- 研究团队
- 检索范式
- GDR：记忆机制的双刃剑效应
- GDR：应用场景

01 研究团队

冯少雄，现负责小红书社区搜索精排LTR、曾负责向量召回（个性化和长尾）。博士毕业于北京理工大学，在 ICLR、AAAI、ACL、EMNLP、NAACL 等机器学习/自然语言处理领域会议/期刊上发表数篇论文。主要研究方向为大语言模型推理评测、生成式检索、开放域对话生成等。

李易为，现博士就读于北京理工大学，小红书社区搜索研究实习生，在 ICLR、ACL、EMNLP、NAACL、AAAI、NeurIPS 等机器学习/自然语言处理领域会议/期刊上发表数篇论文，主要研究方向为大语言模型、开放域对话生成等。

袁沛文，现博士就读于北京理工大学，小红书社区搜索研究实习生，在 NeurIPS、ICLR、ACL、AAAI 、EACL 等发表多篇一作论文。主要研究方向为大语言模型推理与评测、信息检索。

王星霖，现博士就读于北京理工大学，小红书社区搜索研究实习生，在 ACL、EACL 等发表数篇一作论文。主要研究方向为大语言模型推理与评测、信息检索。

02 检索范式


检索范式
稀疏检索
TF-IDF、BM25
扩展：混合检索
密集检索
DSSM
扩展：难负样本挖掘、后交互、知识蒸馏、对比学习
生成式检索
类型一
查询改写、查询结果摘要
类型二
GDR，生成结果id
类型三
ChatGPT (RAG)，生成结果content

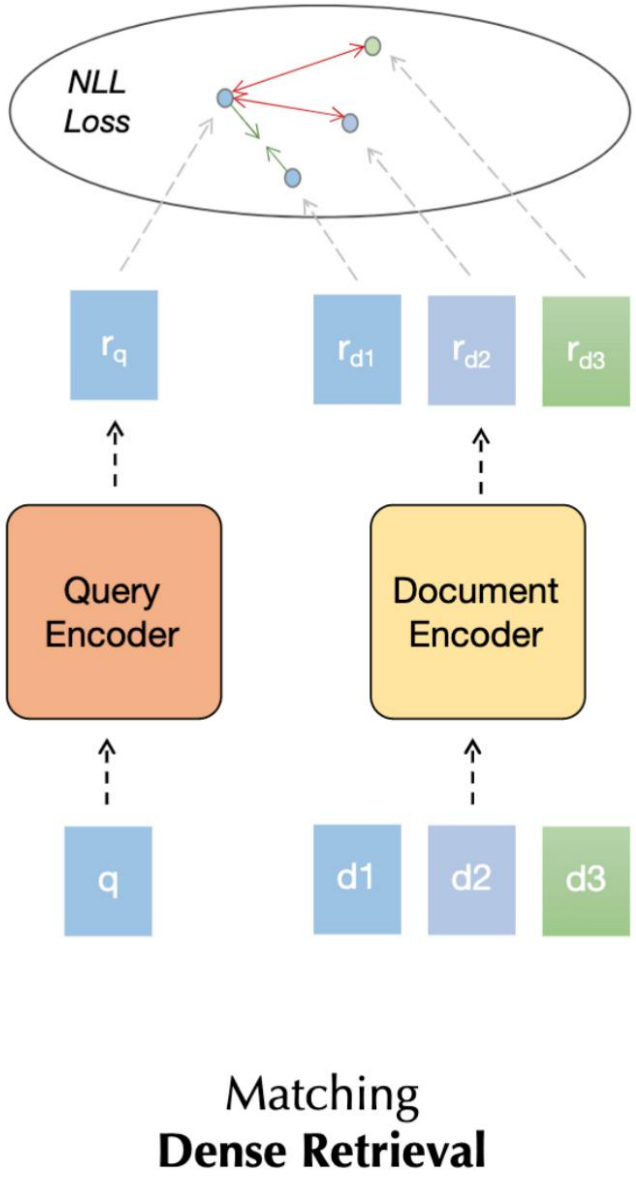
03 GDR：记忆机制的双刃剑效应

小红书

Generative Dense Retrieval: Memory Can Be a Burden (EACL 2024 Oral)

Peiwen Yuan*, Xinglin Wang*, Shaoxiong Feng, Boyuan Pan, Yiwei Li, Heda Wang, Xupeng Miao, Kan Li

Beijing Institute of Technology, Xiaohongshu Inc, Carnegie Mellon University



DR：密集检索[1,2]

利用**语义向量匹配机制**，在各类业务场景中已大规模落地

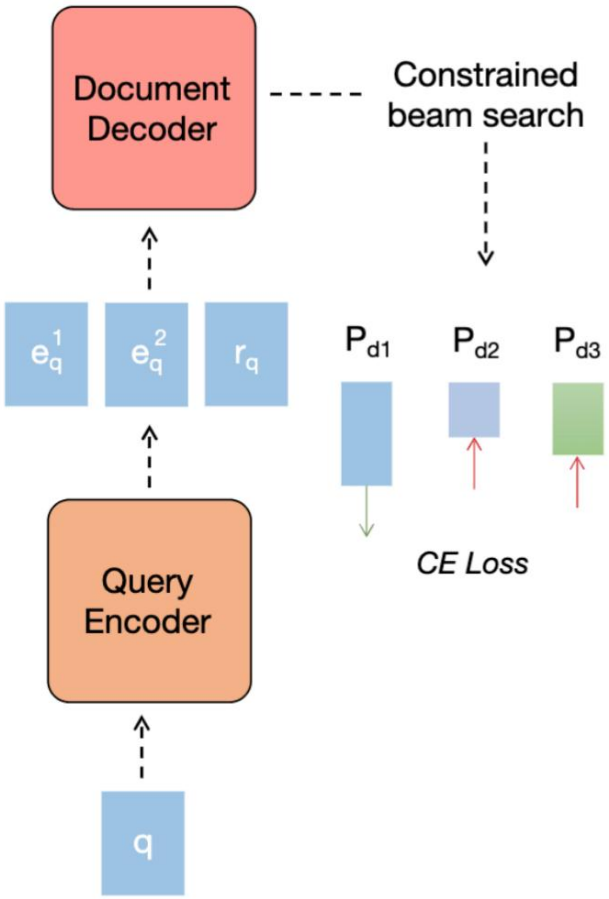
优点：

- 具有较低的时延、可接受的在线计算开销
- 当候选文档库动态更新（增改删）时，可高效更新索引
- 能直接看到文档，捕捉其细粒度特征

不足：

- “查询-候选文档”间的交互仅通过语义向量在欧氏空间进行，缺乏深度交互
- 查询和文档间具有一对多的特性，候选文档间的语义关联可能较远，而查询想召回所有候选文档则要求它们在语义空间接近，产生了矛盾

[1] Karpukhin V, Oğuz B, Min S, et al. Dense passage retrieval for open-domain question answering[J]. arXiv preprint arXiv:2004.04906, 2020.
[2] Zhang H, Gong Y, Shen Y, et al. Adversarial retriever-ranker for dense text retrieval[J]. arXiv preprint arXiv:2110.03611, 2021.



Memorizing
Generative Retrieval

GR：生成式检索[3,4]

利用**记忆机制**，在小规模候选文档场景下展现了优于密集检索的的召回性能

优点：

- 能够以模型参数为载体记忆候选文档，在解码过程中隐式实现查询和候选文档的深度交互
- 文档被赋予独立的Identifier，避免了密集检索面临的候选文档间语义互斥问题
- 能够有效利用自回归预训练语言模型的语义理解能力

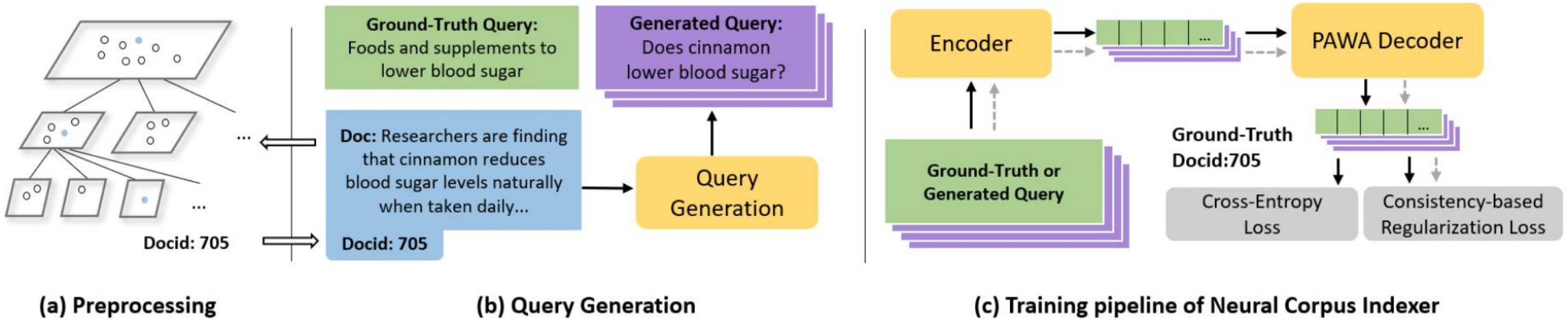
不足：

- ？

[3] Tay Y, Tran V, Dehghani M, et al. Transformer memory as a differentiable search index[J]. NeurIPS, 2022, 35: 21831–21843.
[4] Wang Y, Hou Y, Wang H, et al. A neural corpus indexer for document retrieval[J]. NeurIPS, 2022, 35: 25600–25614.

生成式检索面临的问题 1：对候选文档的细粒度特征有效记忆不足

生成式检索训练流程



通过对每个候选文档构建多条查询来表示整个候选文档，让模型进行记忆

↓

事实上模型不能真正看到整个候选文档（查询集只覆盖了文档的部分内容，查询集未整体作为输入）

↓

对候选文档中具体的、细粒度的信息无法有效（区分性）记忆

生成式检索面临的问题 1：对候选文档的细粒度特征有效记忆不足

小红书

实验设置： 1

对比基线： DR模型 - AR2[5]　　GR模型 - NCI[4]

方法：

step1. 对候选文档，通过BERT获得语义向量，然后k-means聚类获得候选文档的层次化ID（如：3-6-2-8-9-6）

step2. 分别统计AR2和NCI召回的候选文档在第几位与应召回的候选文档发生了差异

（如：NCI召回的候选文档ID是3-6-2-8-9-6，应召回的候选文档ID是3-6-2-4-2-8，则在第4位发生解码错误）

2

Model	Error Rate of the i^{th} Position					
	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	6^{th}
NCI	**1.09**	**1.75**	**1.86**	5.77	14.91	12.66
AR2	1.19	1.77	2.11	**5.44**	**8.03**	**3.05**

Table 1: Error rate (%) on the i^{th} position when decoding document identifiers. See Appendix A.1 for the detailed calculation method.

3

结论：

相比于密集检索模型（AR2），生成式检索模型（NCI）在粗粒度语义匹配时出错率更低，在细粒度方面则更高。

印证了*生成式检索模型对候选文档的细粒度特征有效记忆不足*的论点。

DataFun.

[5] Zhang H, Gong Y, Shen Y, et al. Adversarial retriever–ranker for dense text retrieval[J]. arXiv preprint arXiv:2110.03611, 2021.

生成式检索面临的问题 2：候选文档规模增大会导致性能显著下降

1

模型参数有限

↓

模型记忆容量有限

↓

候选文档容量超过模型记忆容量时，记忆出现瓶颈

↓

模型召回性能下降

2

实验设置：

方法：

step1. 构建size为334K的候选文档集合，训练NCI模型对该候选文档集合进行记忆，并测量其在该候选文档集合下的表现（334K-334K）

step2. 构建size为1M的候选文档集合，训练NCI模型，并测量其在该候选文档集合下的表现（1M-1M），和在334K候选文档集上的表现（1M-334K）

3

Settings	NCI	AR2
	R@1/100	R@1/100
334K-334K	14.7 - / 65.5 -	21.2 - / 69.0 -
1M-1M	11.1↓3.6 / 54.5↓11.0	20.3↓0.9 / 66.2↓2.8
1M-334K	12.3↓2.4 / 59.8↓5.7	21.2 - / 69.0 -

Table 2: Performance of NCI and AR2 on NQ validation set with different settings. For setting $x - y$, x denotes the training corpus size and y denotes the candidate corpus size during the inference phase. AR2 is only trained on the training set, thus is independent of x.

4

结论：

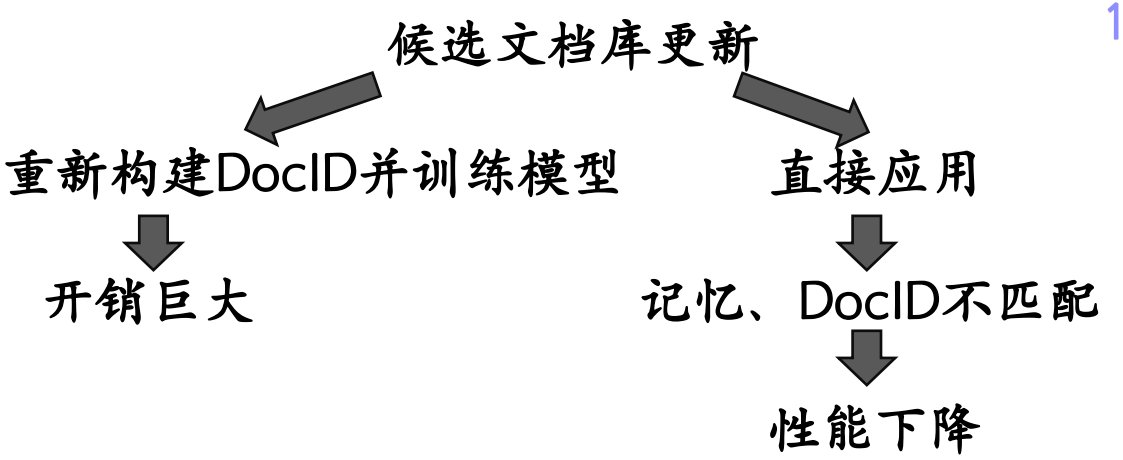
- 334K-334K与1M-334K比较：由于要额外记忆666K候选文档导致NCI的R@100下降5.7
- 1M-1M与1M-334K比较：更多的负例文档导致了R@100额外下降5.3

候选文档规模增大会导致性能显著下降

[6] Pradeep R, Hui K, Gupta J, et al. How does generative retrieval scale to millions of passages?[J]. arXiv preprint arXiv:2305.11841, 2023.

生成式检索面临的问题 3: 候选文档更新时可扩展差

1



2

实验设置:

方法:

step1. 将NQ 数据集的查询均分为两份，各自对应的候选文档集合为Set A和Set B，用Set A训练NCI模型，并测量其在该候选文档集合对应查询上的表现（Set A-Set A）

step2. 将候选集Set A更新为Set A+Set B，不重新训练模型，分别测试其在Set A对应的查询集（All-Set A）和Set B对应的查询集上的表现（All-Set B）

3

$\mathcal{D}_l$	$\mathcal{S}_{val}$	NCI	
		Acc@100	R@100
Set A	Set A	90.7 -	71.2 -
All	Set A	80.7↓10.0	52.9↓18.3
All	Set B	56.5↓34.2	27.7↓43.5

4

结论:

- Set A-Set A与All-Set A比较：由于候选集扩充导致的记忆、DocID不匹配问题导致R@100下降18.3
- All-Set A与All-Set B比较：由于模型没有对于Set B的文档进行记忆，因此测试Set B对应的查询时导致了更加严重的25.2的R@100下降

凭借记忆机制，在较小的记忆需求下能展现良好召回效果，时延较高

不限制候选文档数量，时延可接受，可扩展性强，缺乏查询与文档间的深交互，召回性能理论上无法达到最优

GR：生成式检索　记忆机制

不足：

- 自回归解码范式导致时延大，计算开销大
- 不能看到整个文档，细粒度特征记忆不足
- 候选文档库变化时需要训练模型以修改记忆
- 依赖模型记忆所有候选文档，在候选文档数量增大时表现显著下滑

优点：

- 能够以模型参数为载体记忆候选文档，在解码过程中隐式实现查询和候选文档的深度交互
- 文档被赋予独立的Identifier，避免了密集检索面临的候选文档间语义互斥问题
- 能够有效利用自回归预训练语言模型的语义理解能力

DR：密集检索　语义向量匹配机制

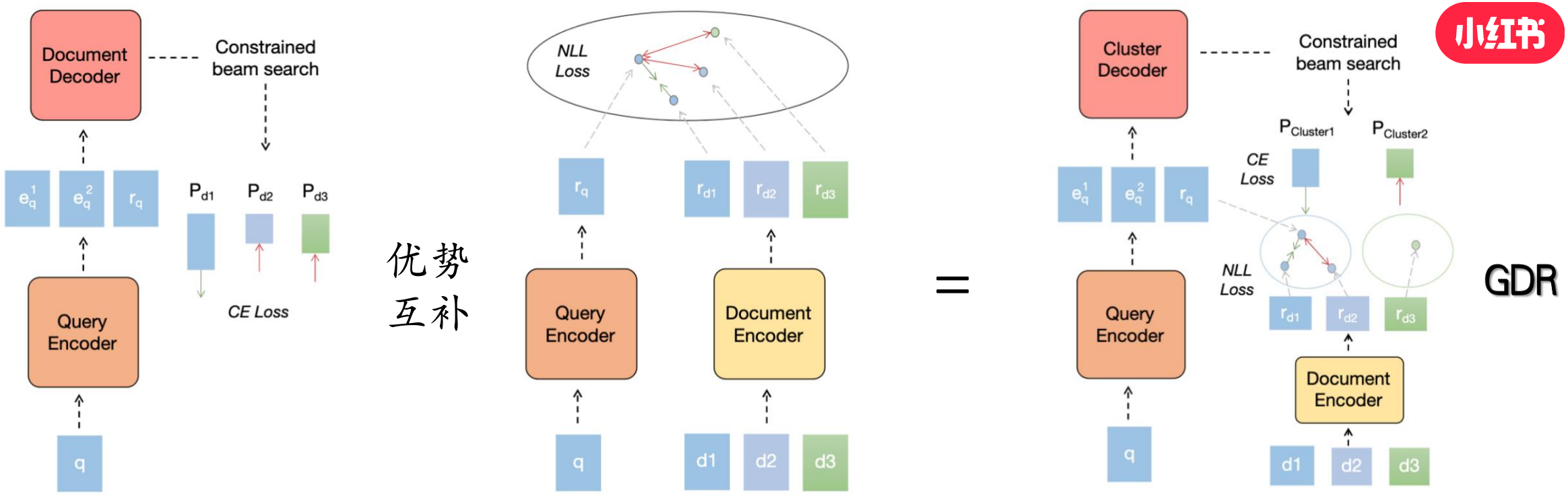
优点：

- 具有较低的时延，可接受的计算开销
- 能直接看到整个文档，捕捉其细粒度特征
- 当候选文档库动态变化时，可高效更新索引
- 当文档部分特征发生变化时，可实现索引快速更新

不足：

- “查询-候选文档”间的交互仅通过语义向量在欧氏空间进行，缺乏深度交互
- 查询和文档间具有一对多的特性，候选文档间的语义关联可能较远，而查询想召回所有候选文档则要求它们在语义空间接近，产生了矛盾

强互补

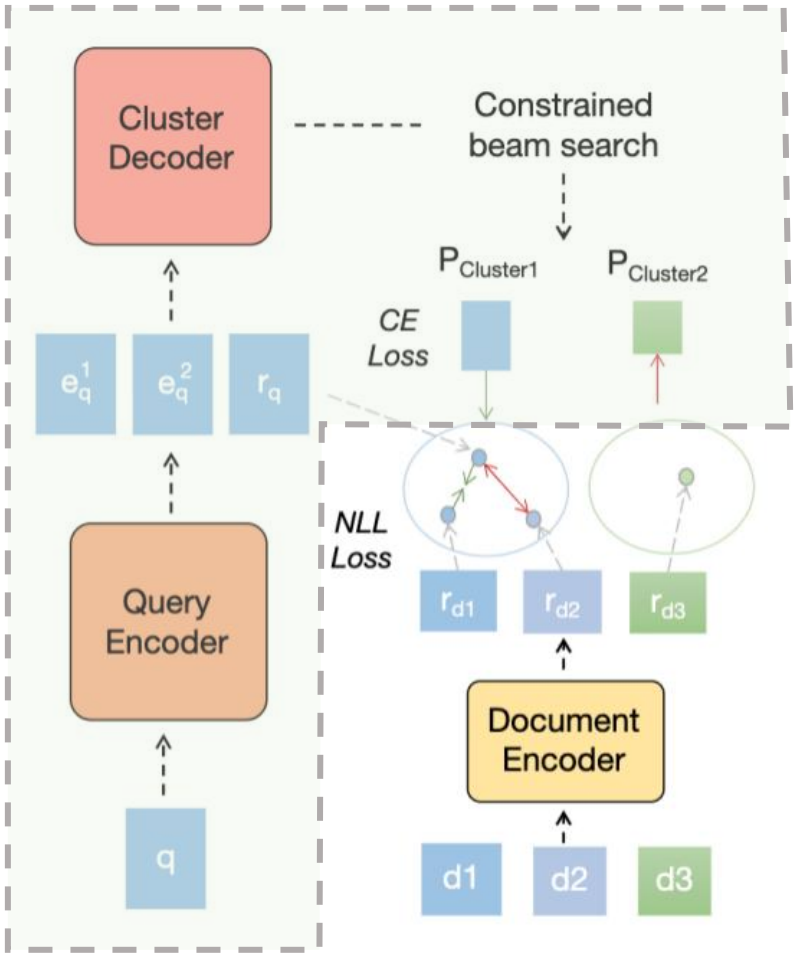


将记忆机制和匹配机制进行层次化结合，各发挥所长

粗粒度映射：记忆机制　　查询-->候选文档簇

细粒度映射：匹配机制　　候选文档簇-->候选文档

GDR方法：粗粒度映射



粗粒度映射：查询-->候选文档簇　　记忆机制

$$f_{inter} : q \rightarrow \mathrm{CID}^{1:k}$$

采用Encodeer-Decoder架构，计算给定查询下候选文档簇概率：

$$p(\mathrm{CID}^i | e_q, r_q, \theta) = \prod_{j=1}^{|\mathrm{CID}^i|} p(\mathrm{CID}^i_j | e_q, r_q, \mathrm{CID}^i_{<j}, \theta)$$

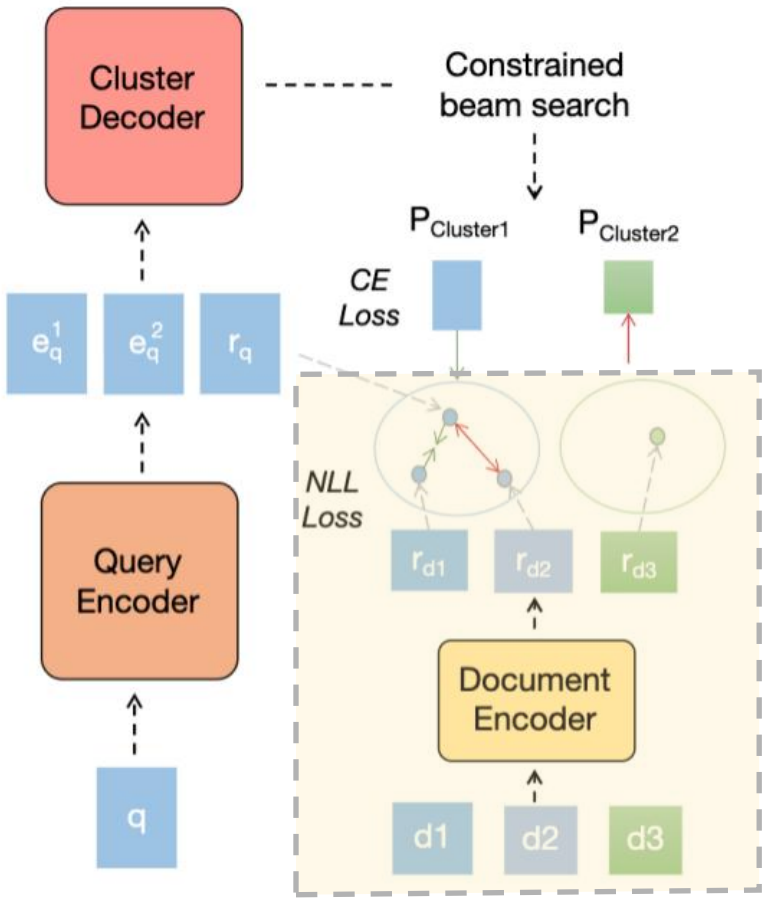
该概率记为簇间映射分数

$$S_{inter}(q, \mathrm{CID}^i) = p(\mathrm{CID}^i | e_q, r_q, \theta)$$

采用交叉熵损失函数：

$$\mathcal{L}_{Inter} = -\log p(\mathrm{CID}(d^+) | E_Q(q), \theta_{D_C})$$

GDR方法：细粒度映射



细粒度映射：候选文档簇-->候选文档　　匹配机制

$$f_{intra} : \mathrm{CID}^{1:k} \rightarrow d^{1:k}$$

引入文档Encodeer，计算给定查询与候选文档簇中文档的匹配度，并记为簇内映射分数

$$S_{intra}(q, d^i) = \mathrm{Sigmoid}(\mathrm{sim}(r_q, r_d^i))$$

采用负对数似然损失函数：

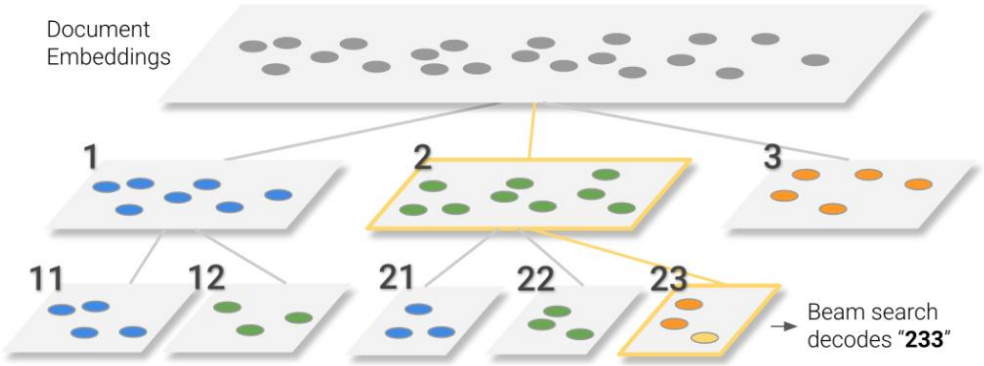
$$\mathcal{L}_{Intra} = -\log \frac{e^{sim(q,d^+)}}{e^{sim(q,d^+)} + \sum_i^n e^{sim(q,d_i^-)}}$$

最终每个候选文档的综合得分为：

$$S_{overall}(q, d^i) = S_{inter}(q, \mathrm{CID}(d^i)) + \beta * S_{intra}(q, d^i)$$

GDR方法：记忆友好的文档簇ID构建

文档簇ID层级结构的构建至关重要



1

传统的文档ID构建：

文档——>BERT——>文档vector——>K-means——>DocID

2

不足：

- 无法控制产生的ID数量，来避免模型记忆过多内容 ✓
- BERT对文档的语义编码结果可能不适合检索任务 ？

3

$$O_{pre} = \frac{1}{|S_{val}|} \sum_{q \in S_{val}} \frac{1}{|\mathcal{D}_q|^2} \sum_{i=1}^{|\mathcal{D}_q|} \sum_{j=1}^{|\mathcal{D}_q|} o_{pre}(\mathrm{CID}_q^i, \mathrm{CID}_q^j)$$

$$o_{pre}(s_1, s_2) = |LCP(s_1, s_2)|/|s_1|$$

4

O_{pre}　　BERT：0.516　　Finetune：0.636

GDR方法：记忆友好的文档簇ID构建

Algorithm 1 Generating document cluster identifiers (CIDs).

Require: Corpus $d^{1:|\mathcal{D}_l|}$, Document Encoder E_D, Inter-cluster number k, Intra-cluster number c
Ensure: Document cluster identifiers $\text{CID}^{1:|\mathcal{D}_l|}$

Encode $d^{1:|\mathcal{D}_l|}$ with E_D to obtain document representations $X^{1:|\mathcal{D}_l|}$
function GENERATECIDS($X^{1:N}$)
  $C^{1:k} \leftarrow Kmeans(X^{1:N})$
  $L \leftarrow \varnothing$
  for $i \leftarrow 1, k$ **do**
    $L_{current} \leftarrow [i] * |C^i|$
    if $|C^i| \geq c$ **then**
      $L_{rest} \leftarrow$ GENERATECIDS(C^i)
    else
      $L_{rest} \leftarrow [0] * |C^i|$
    end if
    $L_{cluster} \leftarrow$Concat($L_{current}, L_{rest}$)
    $L \leftarrow L$.Append($L_{cluster}$)
  end for
  ReorderToOriginal($L, X^{1:N}, C^{1:k}$)
  Return L
end function
$\text{CID}^{1:|\mathcal{D}_l|} \leftarrow$ GENERATECIDS($X^{1:|\mathcal{D}_l|}$)

- BERT对文档的语义编码结果可能不适合检索任务

  在训练集上微调BERT，使之能够产生适合检索任务的层次化文档簇ID

- 无法控制产生的ID数量，来避免模型记忆过多内容

  依据模型大小决定期望的文档簇数量，进而决定文档簇树的叶子结点最大容量：

$$c = |\mathcal{D}_l| \ / \ Exp(|\text{CID}|)$$

GDR方法：基于簇的自适应负采样策略

1

在GDR的细粒度映射阶段，与传统密集检索模型相比：
映射的候选集从全集变成了候选簇内的文档

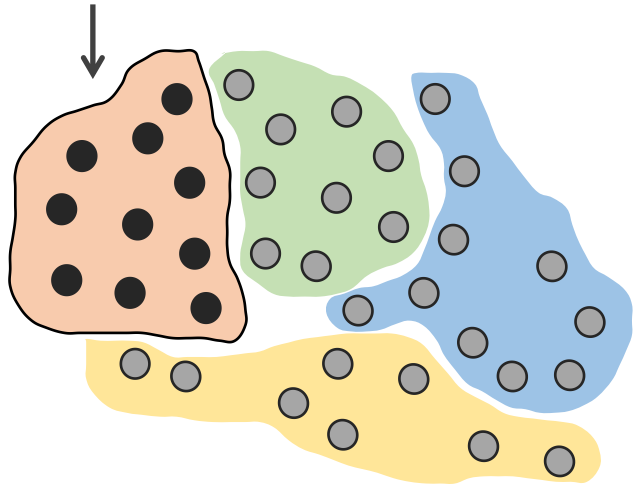
2

为提供更具有区分度的监督信号：
从簇内选择困难负样本，
从簇外选择泛化负样本，
赋予不同的权重

3

$$\mathcal{L}_{Intra} = -\log \frac{e^{\text{sim}(q,d^+)}}{\gamma * \sum_{d \in N_a} e^{\text{sim}(q,d)} + \sum_{d \in N_r} e^{\text{sim}(q,d)}}$$

匹配样本簇



● 困难负样本

● 泛化负样本

实验：对比实验

小红书

实验设置：

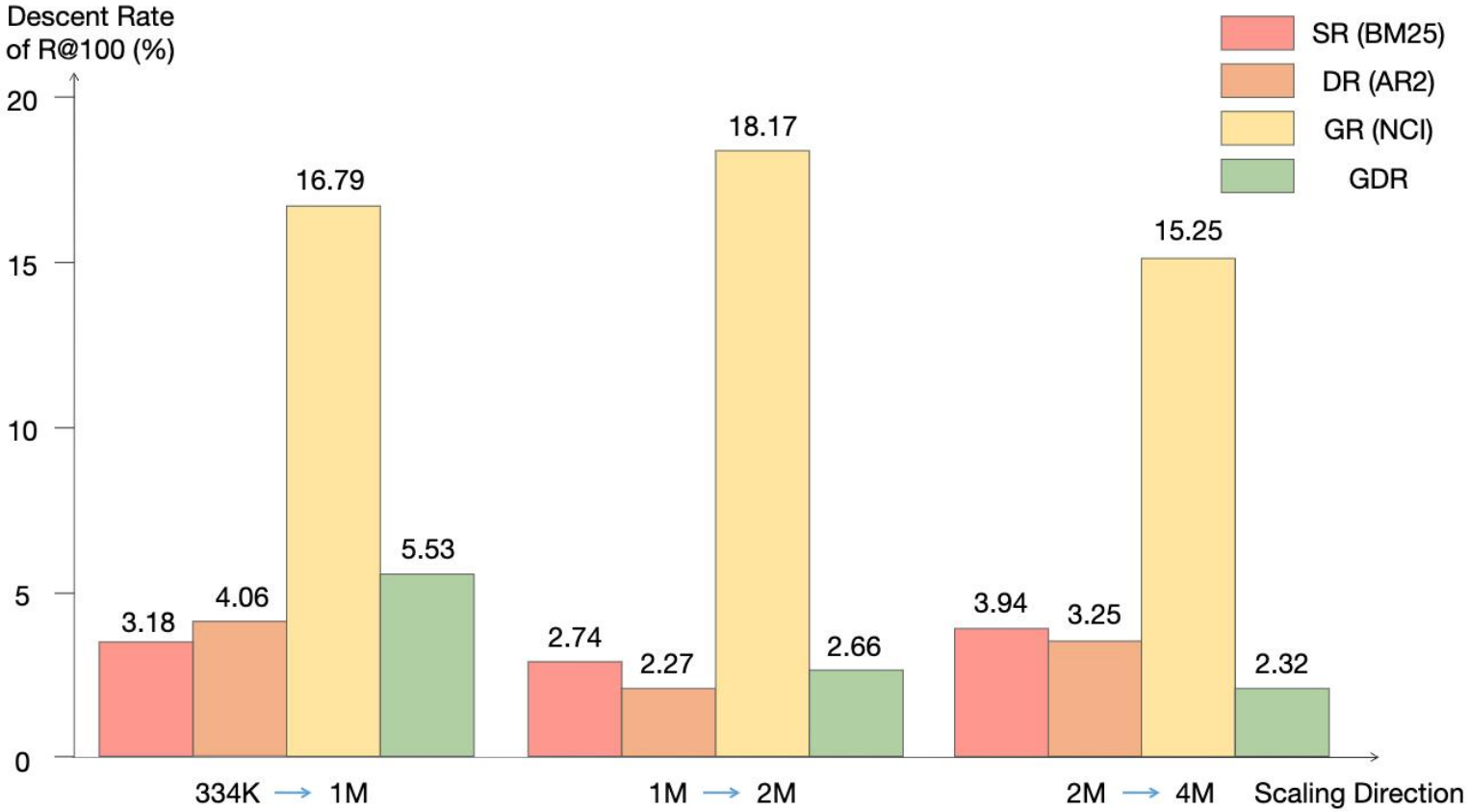
将NQ数据集按照候选集大小划分为NQ334K、NQ1M、NQ2M、NQ4M

Paradigm	Method	NQ334K		NQ1M		NQ2M		NQ4M	
		Acc@20/100	R@20/100	Acc@20/100	R@20/100	Acc@20/100	R@20/100	Acc@20/100	R@20/100
SR	BM25	86.1 / 92.4	56.0 / 75.4	84.0 / 91.0	51.3 / 73.0	82.4 / 89.9	47.5 / 71.0	79.6 / 88.4	42.3 / 68.2
DR	DPR	93.9 / 97.3	49.8 / 60.2	91.5 / 96.3	46.7 / 56.6	90.4 / 95.5	45.2 / 54.9	88.4 / 94.6	42.9 / 52.8
	AR2	**96.3 / 98.6**	57.4 / 69.0	**94.9 / 98.0**	54.7 / 66.2	**94.3 / 97.7**	53.2 / 64.7	**93.4 / 97.2**	51.2 / 62.6
GR	NCI-bert	80.0 / 88.7	49.4 / 65.5	72.0 / 82.6	38.7 / 54.5	63.9 / 76.4	30.2 / 44.6	55.4 / 70.0	25.2 / 37.8
	NCI-ours	88.0 / 94.1	60.0 / 75.6	80.3 / 89.6	50.6 / 66.2	78.2 / 88.6	46.4 / 63.5	77.3 / 87.8	45.2 / 61.0
GDR	GDR-bert	87.5 / 91.2	59.3 / 71.2	84.8 / 88.8	54.8 / 66.0	83.3 / 88.0	51.9 / 64.8	82.1 / 87.7	49.7 / 63.8
	GDR-ours	91.1 / 95.3	**64.6 / 79.6**	88.2 / 93.6	**60.1 / 75.2**	87.4 / 92.8	**57.7 / 73.2**	87.0 / 92.2	**55.2 / 71.5**

结论：

- R@k： **GDR** (GDR-ours) > SR (BM25) > DR (AR2) > GR (NCI)
- Acc@k： DR (AR2) > **GDR** (GDR-ours) > SR (BM25) > GR (NCI)

实验：扩展至更大候选集



结论：

随候选集增大，性能下降幅度：GR>>SR ~ DR ~ GDR

实验：候选文档更新 & 检索效率

$\mathcal{D}_l$	$\mathcal{S}_{val}$	NCI		GDR	
		Acc@100	R@100	Acc@100	R@100
Set A	Set A	90.7 -	71.2 -	94.9 -	77.7 -
All	Set A	80.7↓10.0	52.9↓18.3	93.4↓1.0	75.8↓1.9
All	Set B	56.5↓34.2	27.7↓43.5	86.6↓8.3	66.2↓11.5

新文档加入时，无需训练，可扩展性大幅提升

Method	Latency (ms)	Throughput (queries/s)	Index Refresh (mins)
BM25	56	22.8	2
AR2	35	589.0	5
NCI	232	6.3	-
GDR	195	7.2	7

效率相比于生成式检索提升，距离稀疏、密集检索有差距

实验：Ablation Study

Paradigm	Method	NQ334K		NQ1M		NQ2M		NQ4M	
		Acc@20/100	R@20/100	Acc@20/100	R@20/100	Acc@20/100	R@20/100	Acc@20/100	R@20/100
SR	BM25	86.1 / 92.4	56.0 / 75.4	84.0 / 91.0	51.3 / 73.0	82.4 / 89.9	47.5 / 71.0	79.6 / 88.4	42.3 / 68.2
DR	DPR	93.9 / 97.3	49.8 / 60.2	91.5 / 96.3	46.7 / 56.6	90.4 / 95.5	45.2 / 54.9	88.4 / 94.6	42.9 / 52.8
	AR2	**96.3 / 98.6**	57.4 / 69.0	**94.9 / 98.0**	54.7 / 66.2	**94.3 / 97.7**	53.2 / 64.7	**93.4 / 97.2**	51.2 / 62.6
GR	NCI-bert	80.0 / 88.7	49.4 / 65.5	72.0 / 82.6	38.7 / 54.5	63.9 / 76.4	30.2 / 44.6	55.4 / 70.0	25.2 / 37.8
	NCI-ours	88.0 / 94.1	60.0 / 75.6	80.3 / 89.6	50.6 / 66.2	78.2 / 88.6	46.4 / 63.5	77.3 / 87.8	45.2 / 61.0
GDR	GDR-bert	87.5 / 91.2	59.3 / 71.2	84.8 / 88.8	54.8 / 66.0	83.3 / 88.0	51.9 / 64.8	82.1 / 87.7	49.7 / 63.8
	GDR-ours	91.1 / 95.3	**64.6 / 79.6**	88.2 / 93.6	**60.1 / 75.2**	87.4 / 92.8	**57.7 / 73.2**	87.0 / 92.2	**55.2 / 71.5**

构建记忆友好的文档簇ID ✓

Strategy	Acc@20	Acc@100	R@20	R@100
Random	87.1	91.4	60.8	76.0
BM25	90.2	94.6	63.1	78.5
Cluster-adaptive	**91.1**	**95.3**	**64.6**	**79.6**

基于簇的自适应负采样策略 ✓

β	Acc@20	Acc@100	R@20	R@100
0	70.5	83.9	39.2	59.4
0.5	89.1	93.7	61.9	77.2
1	**91.1**	**95.3**	**64.6**	**79.6**
2	90.9	95.0	64.4	79.5
1e5	90.4	94.8	63.1	77.9

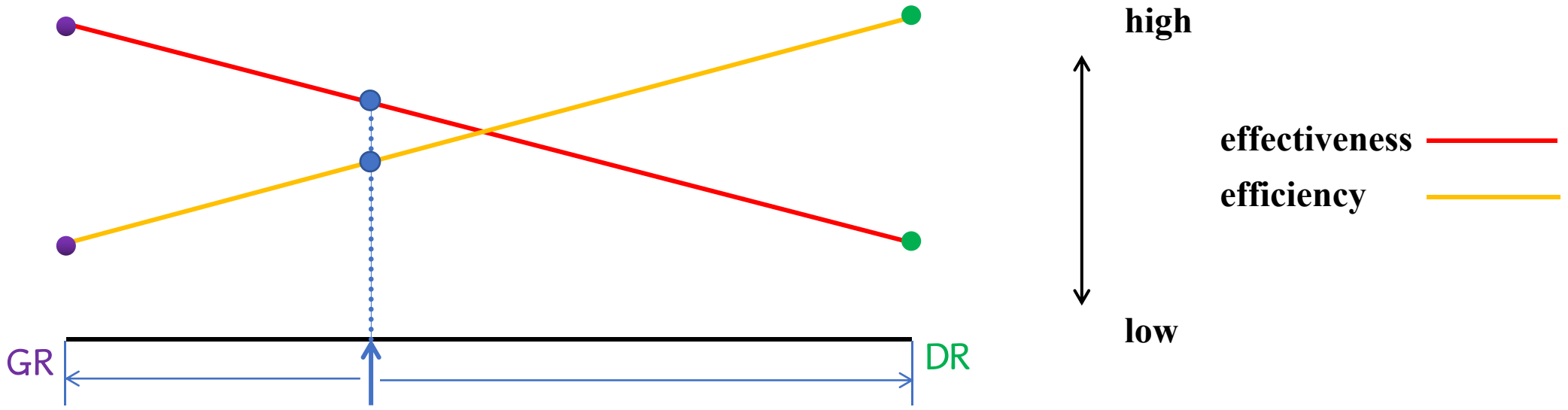
簇内簇间映射分数集成 ✓

总结

主要贡献：

- 对现有的两种检索范式（DR、GR）的优缺点进行了深入分析
- 针对性地设计了GDR范式，实现GR与DR优势互补的效果
- 实验验证分析了GDR的潜力与待解决的问题

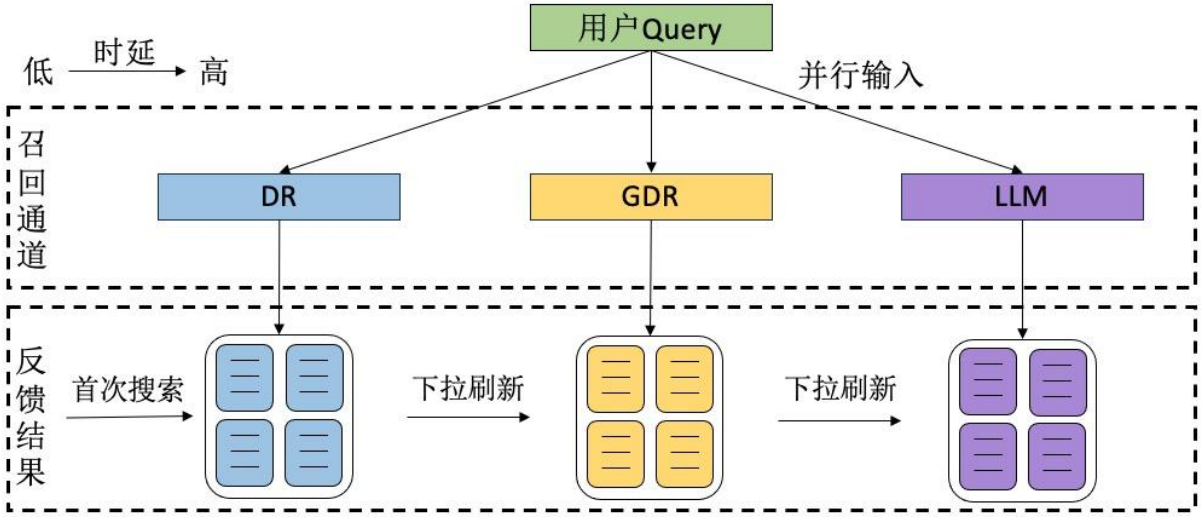
再看GR、DR、GDR



GDR，由CID策略控制效率与效果的平衡

03

GDR：应用场景

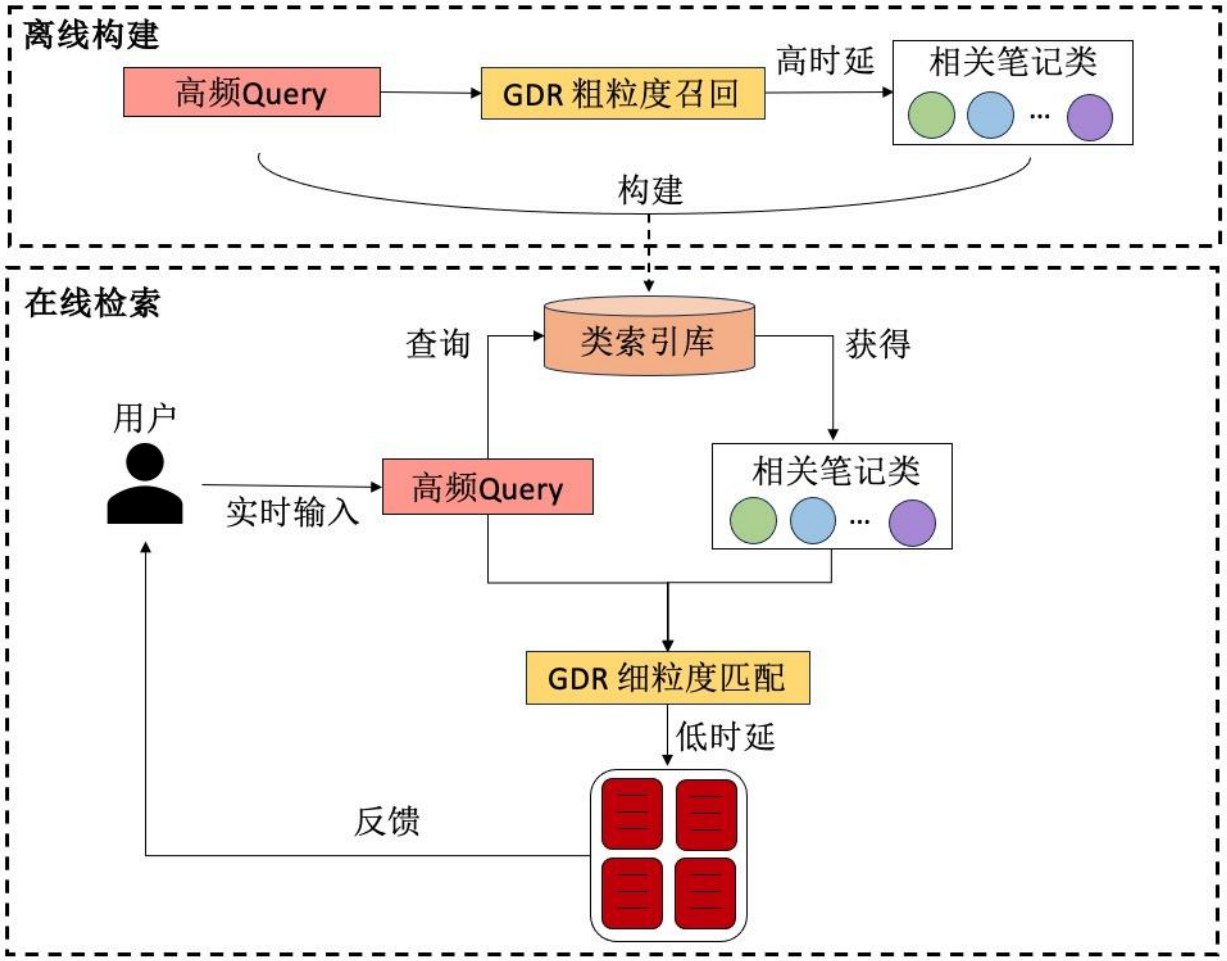


在线流式笔记召回示意图

落地场景一：在线流式笔记召回

对于用户query请求，并发地通过DR(常规、低时延)与GDR（精准、较高时延）进行检索，并流式地将检索结果反馈给用户（首次搜索反馈DR检索结果，下拉刷新反馈GDR检索结果），以在满足用户时延需求的同时提升检索结果的质量。

- 克服GDR较高时延，避免破坏用户体验。
- 提升反馈笔记的整体质量，优化用户体验。
- 针对候选文档库快速更新的工业场景需求，GDR可实现高效的索引更新。
- 增加可接受的计算开销（CID）。

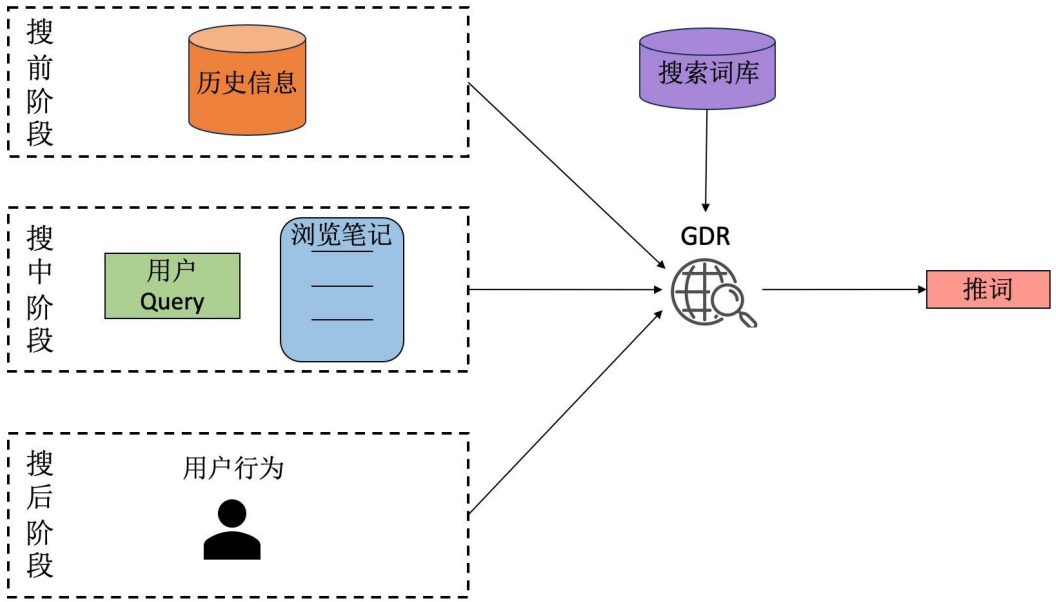


针对高频query的检索优化流程图

落地场景二：针对高频query的检索优化

对于顶头部query，可预先离线通过GDR进行粗粒度相关笔记类召回，维护高频query的相关笔记类索引库，在用户实时提出高频query查询请求时，快速从索引库中匹配相关笔记类，进而通过DR（GDR细粒度检索阶段）在相关笔记类中进行精准的笔记检索，从而在低时延条件下针对高频query实现GDR全流程召回，提升反馈笔记的质量。

- 通过预先构建相关笔记类索引库，降低GDR检索的时延，满足实际用户对结果反馈时延的需求。
- 通过GDR召回的笔记更加精准，优化用户体验。
- 由高频query集中的特点，维护相关笔记类索引库的代价可接受。

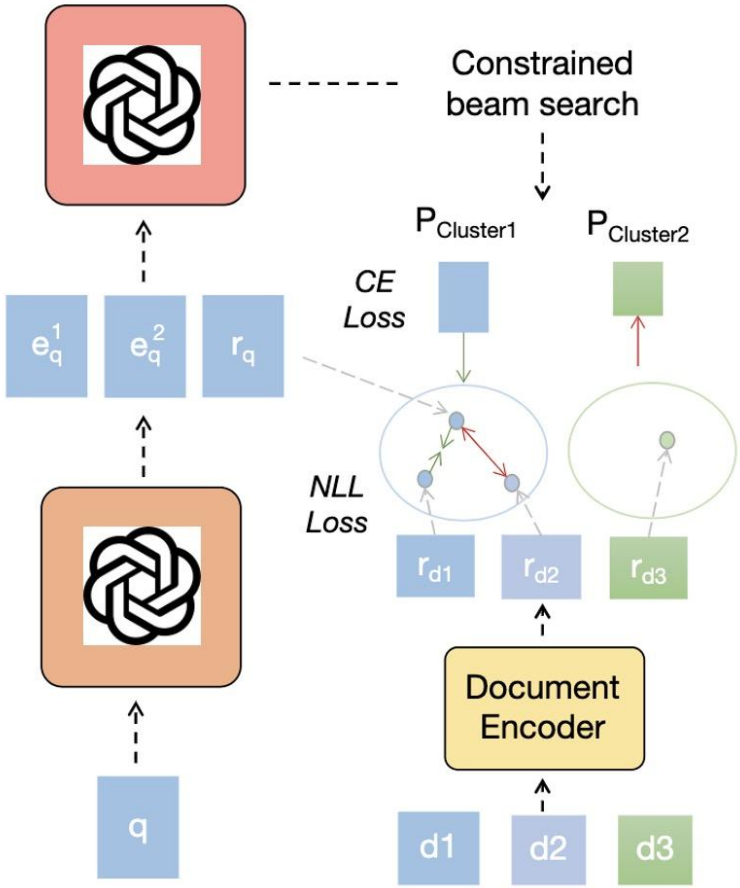


GDR推词场景应用示意图

落地场景三：搜前、搜中、搜后全流程推词

推词对应的候选集数量相对较小（百万级），在GDR实验设定的候选集数量范围中，当前参数量下GDR模型的性能不会受到巨量候选集的影响（例如亿级文档）。
在搜前、搜中、搜后三个阶段，基于用户及历史信息、查询词和浏览笔记，使用GDR可实现更精准的查询词推荐，诸如输入联想、猜你想搜、相关搜索等。

- 借助GDR记忆和匹配能力，提升推词的整体质量，同时避免巨量候选集对GDR检索性能的影响。
- GDR中使用LLMs作为编码器，可利用其强大的理解能力和世界知识，尤其对于长尾查询词。
- 推词业务中部分场景对应的用户时延要求较低，避免GDR较高的时延对用户体验的影响。



GDR+LLM

未来落地场景一：GDR+LLM

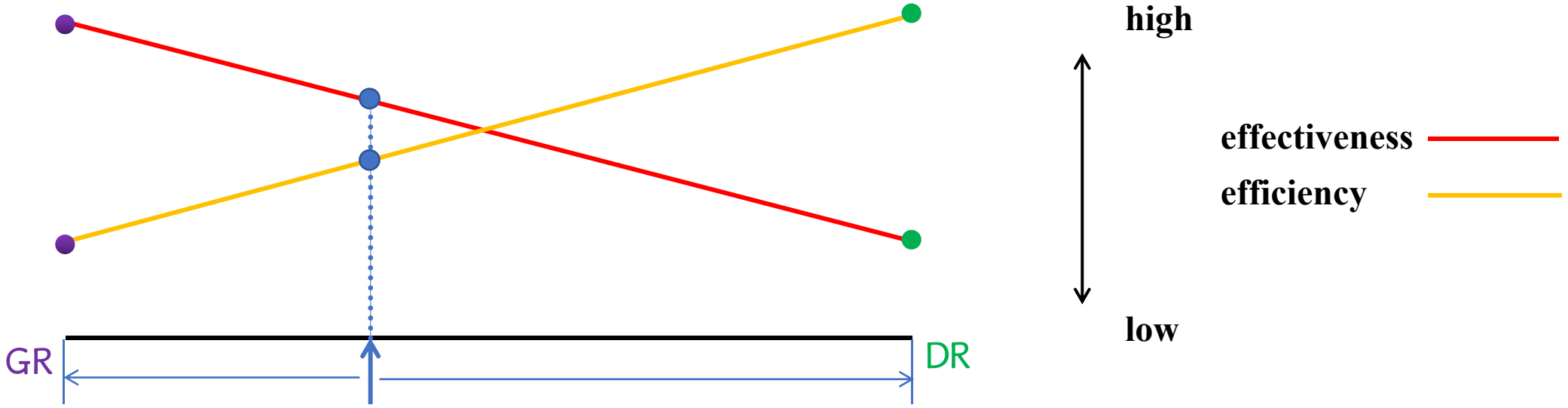
在应对文档库频繁更新的场景时，基于LLM的生成式QA系统需要高频的对LLM在大量新数据上进行微调，其所需的时间、计算资源极高，难以满足现实需求。
我们考虑结合LLM的强大理解能力与GDR自身快速更新的特性进一步增强检索系统的性能，同时满足文档库频繁更新的需求。

- GDR能够对大量新文档快速更新，但缺少LLM的强大query理解能力。
- LLM有强大的query理解能力，但其难以应对文档库的快速更新。
- 结合两者所长能够在满足笔记库快速更新的需求，同时提升检索系统的性能。

未来落地场景二：计算效率自适应的检索策略

计算效率会随着硬件技术的改进而不断提升，在满足基本用户时延的前提下，如何更灵活地分配计算资源、提升检索性能？GDR能够通过CID 构建策略（调整文档簇中包含文档的平均数量）实现效率和性能的“交换”，以在特定时延的需求下最大化利用计算资源提升检索性能。

- 面对不同硬件的计算效率，自适应地调整检索算法的复杂度，以满足时延需求。



GDR，由CID策略控制效率与效果的平衡

THANKS

DataFunSummit # 2024