

Generative Retrieval

Pengjie Ren (任鹏杰)
Shandong University, Qingdao, China

NLPCC 2023

The 12th CCF International Conference on Natural Language Processing and Chinese Computing
Foshan, China, October 12-15, 2023



Slides made by...



Zhongkun Liu (刘中坤)



Wenhao Zhang (张文浩)



Pengjie Ren (任鹏杰)

Generators

Discriminator



Outline

- **Introduction to Retrieval Systems**
- Three Main Challenges of Generative Retrieval
 - Identifier Design
 - Training Strategy
 - Dynamic Corpora
- Open Discussion

Definition

- Information Retrieval (IR) is **finding material** (usually documents) of an **unstructured** nature (usually text) that satisfies an **information need** from within **large collections** (usually stored on computers).

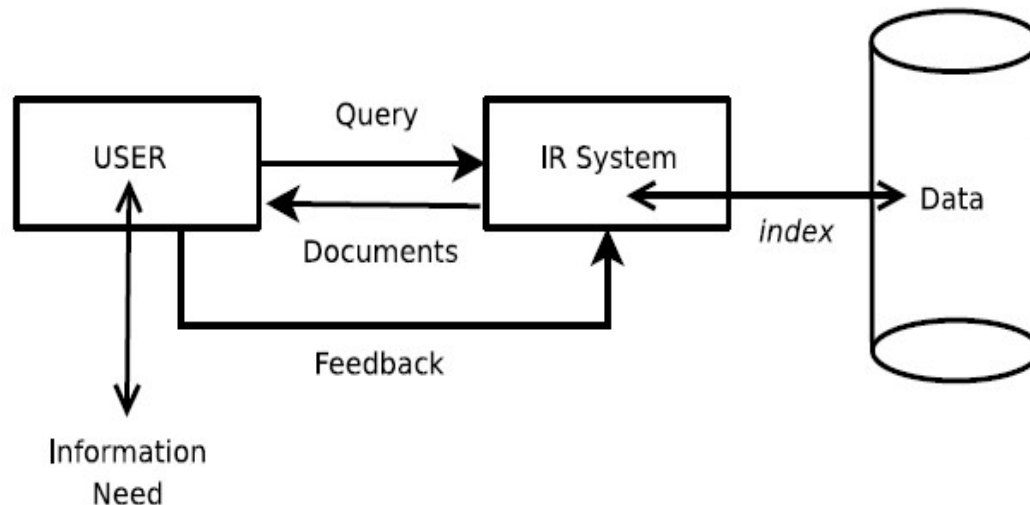


Connecting people to information

s.t. to the **right** people;
with the **right** information;
at the **right** time;
in the **right** place;
in the **right** form.

Information Retrieval Process

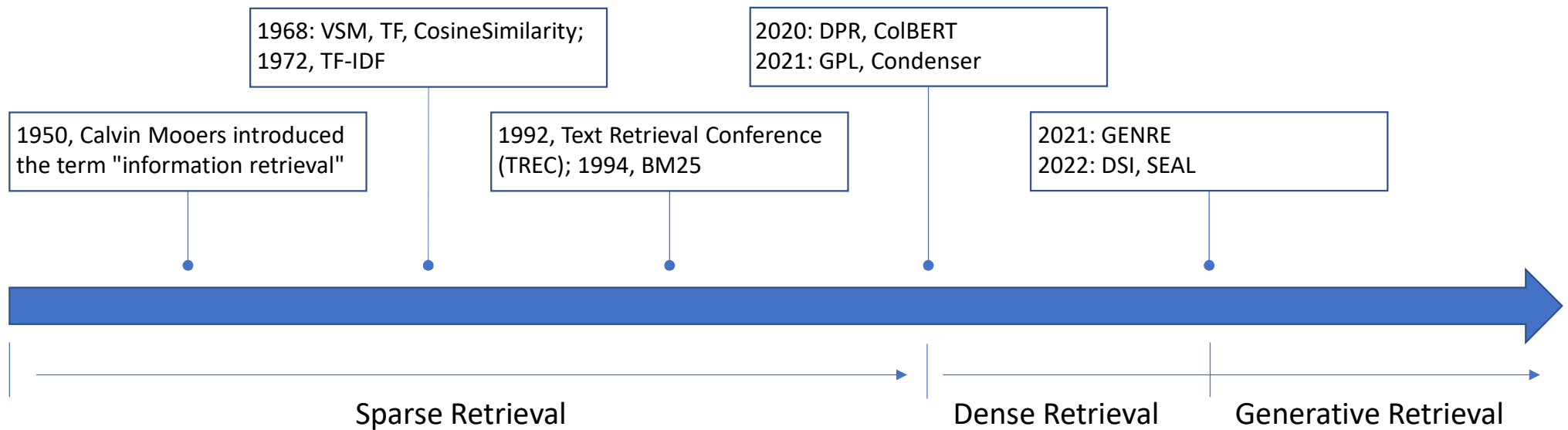
1. Asking a question (how to use the language to get what we want?)
2. Building an answer from known data (how to refer to a given text?)
3. Assessing the answer (does it contain the information we were seeking?)



- This process can loop, the feedback phase allows to improve the quality of the answers.
- The user may not be able to define exactly the query (i.e. characterize his/her information needs).

Trend: Relieve the burden of users.

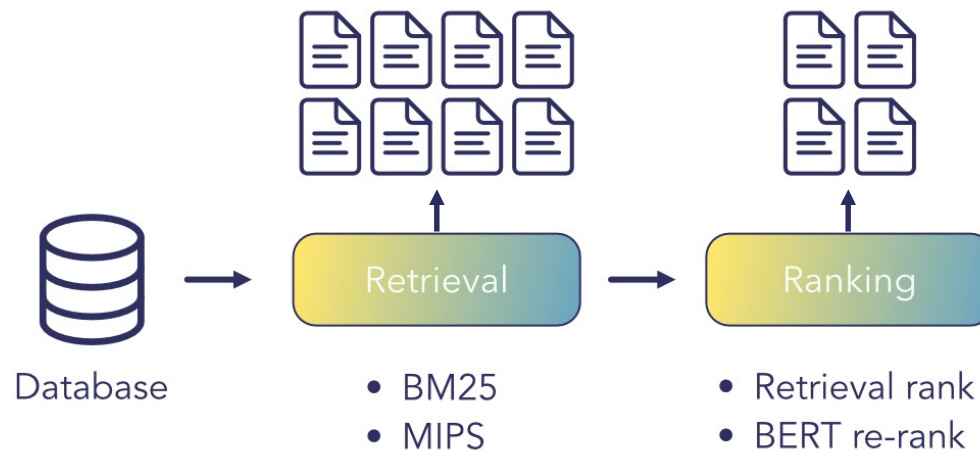
Timeline of IR



- Information retrieval can be traced back to the 1950s.
- Dense retrieval and generative retrieval are popular in recent 3 years due to pre-trained models.

Two-stage IR

- Can we help the user find the right information as fast as possible from a large pool?



- Stage1 retrieval: fast due to offline indexing, focus on recall.
- Stage2 ranking: slow but more precise, focus on precision.

Vocabulary mismatch problem



How big is a tiger?

The **tiger** is the **biggest** member of the Felidae family

Panthera Tigris can reach 390cm nose to tail

False positive

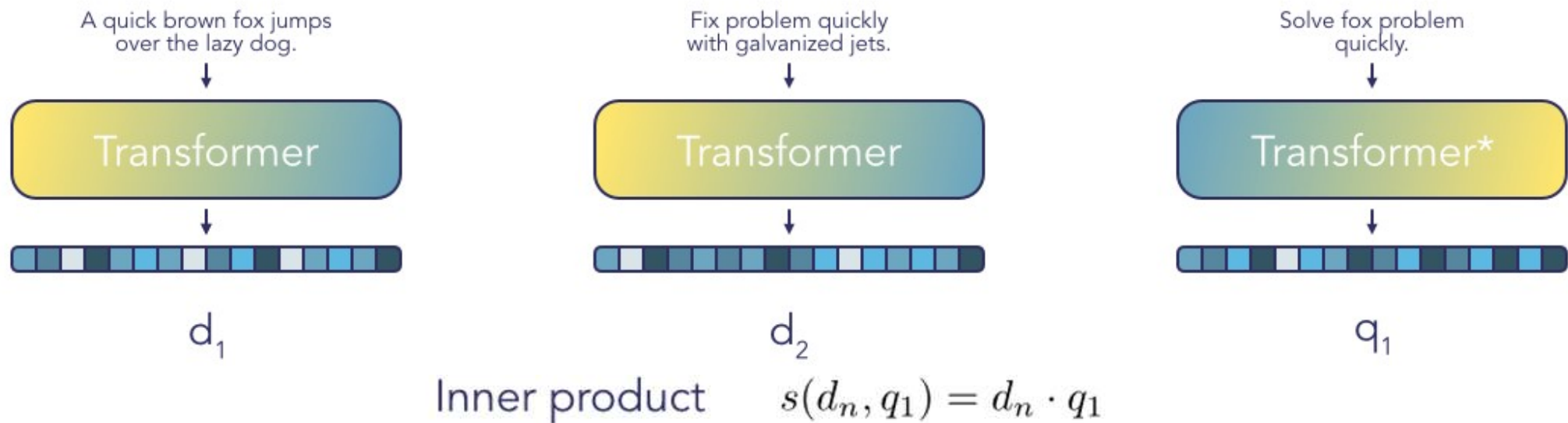
False negative



- Sparse retrieval cannot well handle the difference in semantics.

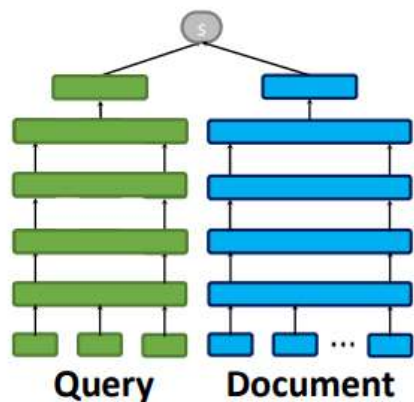
Dense Retrieval

- Dense retrieval allows to represent a text in a form of vector in some vector space with predefined dimension.
- **Semantically close texts** are usually represented by vectors that are **close in a vector space**.



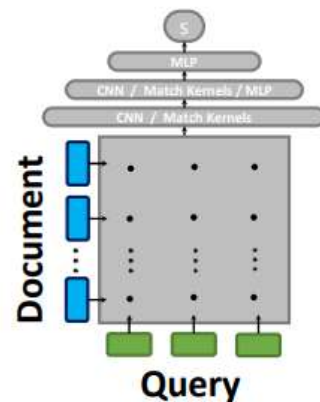
Variants of Dense Retrieval Models

- Aspects:
 - Single-tower vs. dual-tower, i.e., whether to separately process input feature values.
 - Offline indexing vs. online indexing, i.e. whether to utilize pre-computed indexes for score calculations.



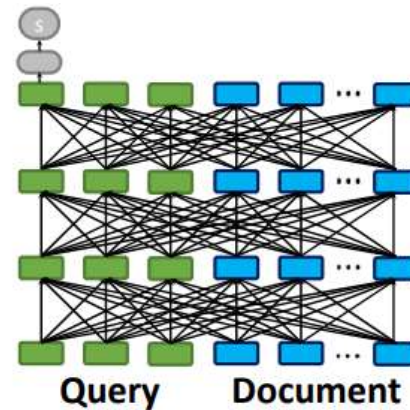
(a) Representation-based Similarity
(e.g., DSSM, SNRM)

- Dual-tower
- Offline indexing



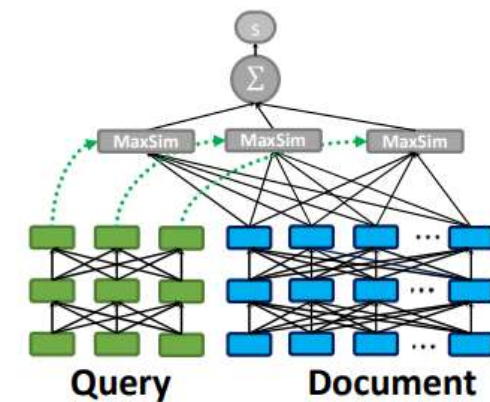
(b) Query-Document Interaction
(e.g., DRMM, KNRM, Conv-KNRM)

- Single-tower
- Online indexing



(c) All-to-all Interaction
(e.g., BERT)

- Dual-tower
- Online indexing



(d) Late Interaction
(i.e., the proposed ColBERT)

- Dual-tower
- Mixed indexing

Domain Adaption of Dense Retrieval Models

- In some cases, dense retrieval models perform even worse than BM25.

Method \ Dataset	FiQA	SciFact	BioASQ	TRECC.	CQADup.	Robust04	Avg.
<i>Zero-Shot Models</i>							
MS MARCO	26.7	57.1	52.9	66.1	29.6	39.0	45.2
PAQ	15.2	53.3	44.0	23.8	24.5	31.9	32.1
PAQ + MS MARCO	26.7	57.6	53.8	63.4	30.6	37.2	44.9
TSDAE _{MS MARCO}	26.7	55.5	51.4	65.6	30.5	36.6	44.4
BM25	23.9	66.1	70.7	60.1	31.5	38.7	48.5

- BM25's scoring function is **domain-independent**, relying on statistical properties of the data.
- Dense retrieval models rely on pre-trained embeddings or complex neural network architectures, which require extensive training on **domain-specific** data to perform well.

Wang, K., Thakur, N., Reimers, N., & Gurevych, I. (2022). GPL: Generative Pseudo Labeling for Unsupervised Domain Adaptation of Dense Retrieval. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, (pp. 2345–2360).

Comparison

Sparse Retrieval

Dense Retrieval

- Search
- Document Representation
 - Dimension
 - Information
- Training
- Domain sensitivity

By keyword.

Term dictionary cardinality

Mostly zeroes in vector.

Not need.

Low.

By semantic.

Much smaller.

Mostly non-zeroes.

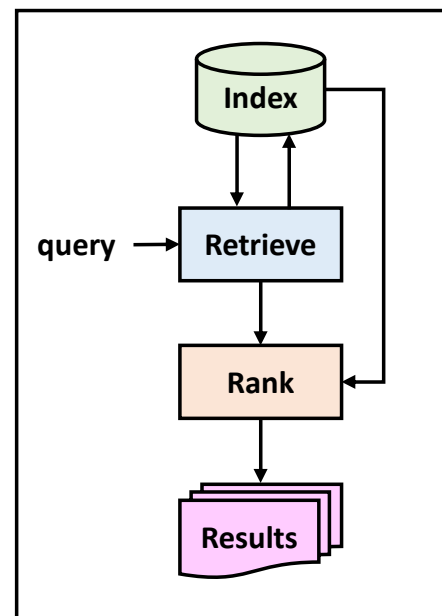
Need.

High.

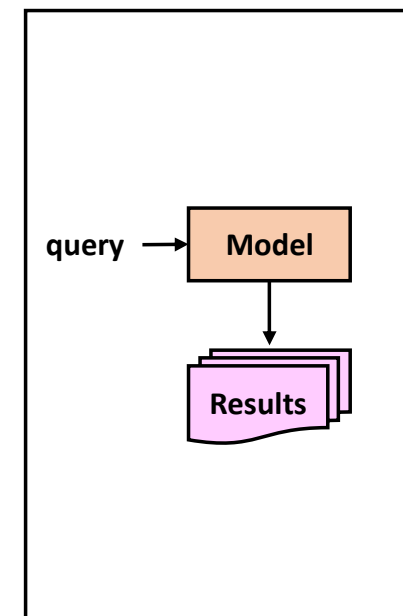
"**retrieve-then-rank**" paradigm

Generative Retrieval

- Generative retrieval is meant to replace the long-lived "**retrieve-then-rank**" paradigm by collapsing the indexing, retrieval, and ranking components of traditional Information Retrieval (IR) systems into **a single unified model**.
- Under the generative retrieval framework, **indexing** is replaced with **model training**, while **retrieval and ranking** are replaced with **model inference**.



Retrieve-then-rank



Generative retrieval

No explicit retriever?



Key question:

“Can we use pre-trained language models to act as **knowledge storage**?”

- Transformers should intuitively serve as a good associative **memory store** or **search index**.
- Generative model can involve both the **semantic information** and **corresponding document identifiers** through pre-training, thereby **replacing traditional static indexes**.

The interpretation of a large Transformer model as a **memory store** have been investigated in prior work.



No explicit retriever?

They train T5 models to **retrieve facts** that are encoded within the parameters of the model during pretraining.

How Much Knowledge Can You Pack Into the Parameters of a Language Model?

Adam Roberts*
Google
adarob@google.com

Colin Raffel*
Google
craffel@gmail.com

Noam Shazeer
Google
noam@google.com

Abstract

It has recently been observed that neural language models trained on unstructured text can implicitly store and retrieve knowledge using natural language queries. In this short paper, we measure the practical utility of this approach by fine-tuning pre-trained models to answer questions *without access to any external context or knowledge*. We show that this approach scales with model size and performs competitively with open-domain systems that explicitly retrieve answers from an external knowledge source when answering questions. To facilitate reproducibility and future work, we release our code and trained models.¹

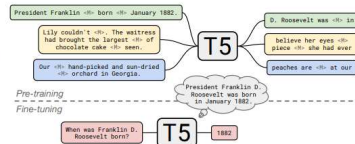


Figure 1: T5 is pre-trained to fill in dropped-out spans of text (denoted by $\langle \text{M} \rangle$) from documents in a large, unstructured text corpus. We fine-tune T5 to answer questions without inputting any additional information or context. This forces T5 to answer questions based on “knowledge” that it internalized during pre-training.

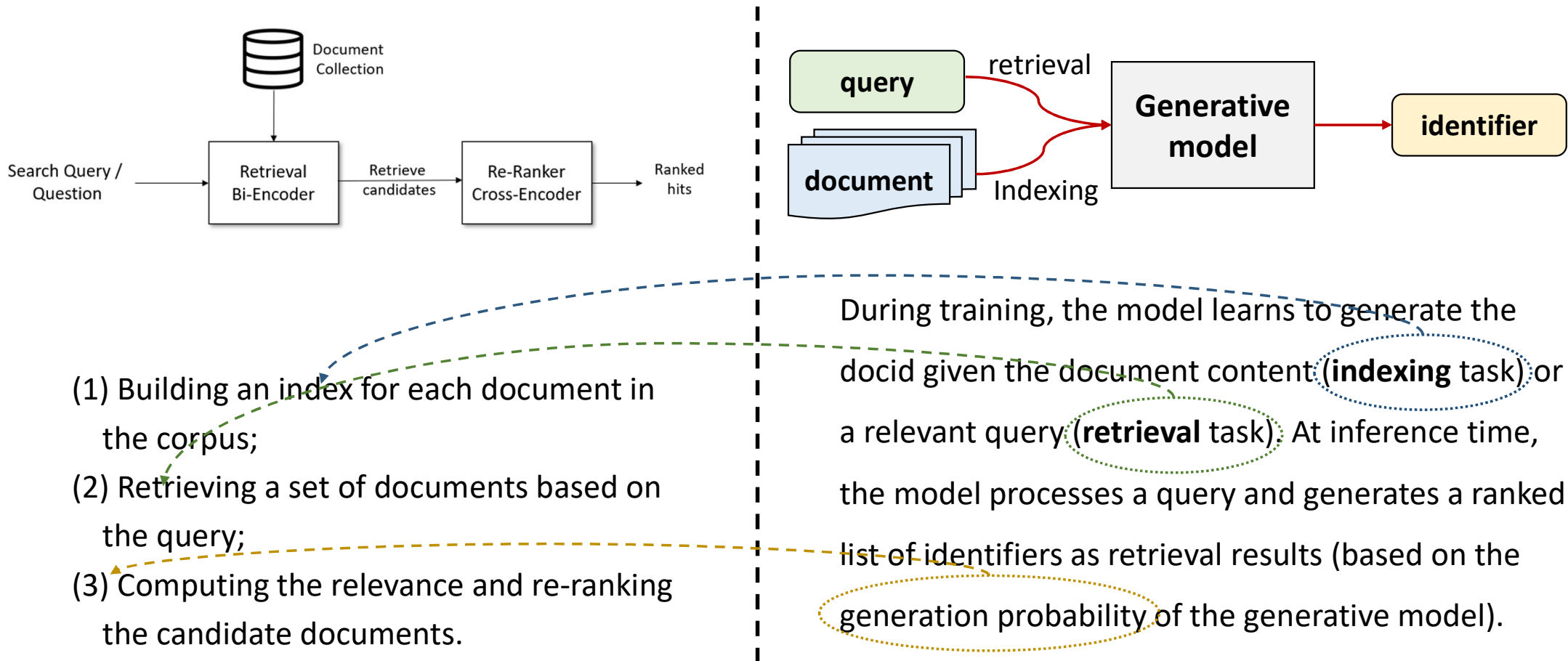
- The method of fine-tuning pretrained models to answer questions **without access to any external context or knowledge** is competitive compared to the method of **explicitly retrieving from external knowledge sources**.

	NQ	WQ	TQA	
			dev	test
Chen et al. (2017)	–	20.7	–	–
Lee et al. (2019)	33.3	36.4	47.1	–
Min et al. (2019a)	28.1	–	50.9	–
Min et al. (2019b)	31.8	31.6	55.4	–
Asai et al. (2019)	32.6	–	–	–
Ling et al. (2020)	–	–	35.7	–
Guu et al. (2020)	40.4	40.7	–	–
Férvy et al. (2020)	–	–	43.2	53.4
Karpukhin et al. (2020)	41.5	42.4	57.9	–
T5-Base	25.9	27.9	23.8	29.1
T5-Large	28.5	30.6	28.7	35.9
T5-3B	30.4	33.6	35.1	43.4
T5-11B	32.6	37.2	42.3	50.1
T5-11B + SSM	34.8	40.8	51.0	60.5
T5.1.1-Base	25.7	28.2	24.2	30.6
T5.1.1-Large	27.3	29.5	28.5	37.2
T5.1.1-XL	29.5	32.4	36.0	45.1
T5.1.1-XXL	32.8	35.6	42.9	52.5
T5.1.1-XXL + SSM	35.2	42.8	51.9	61.6

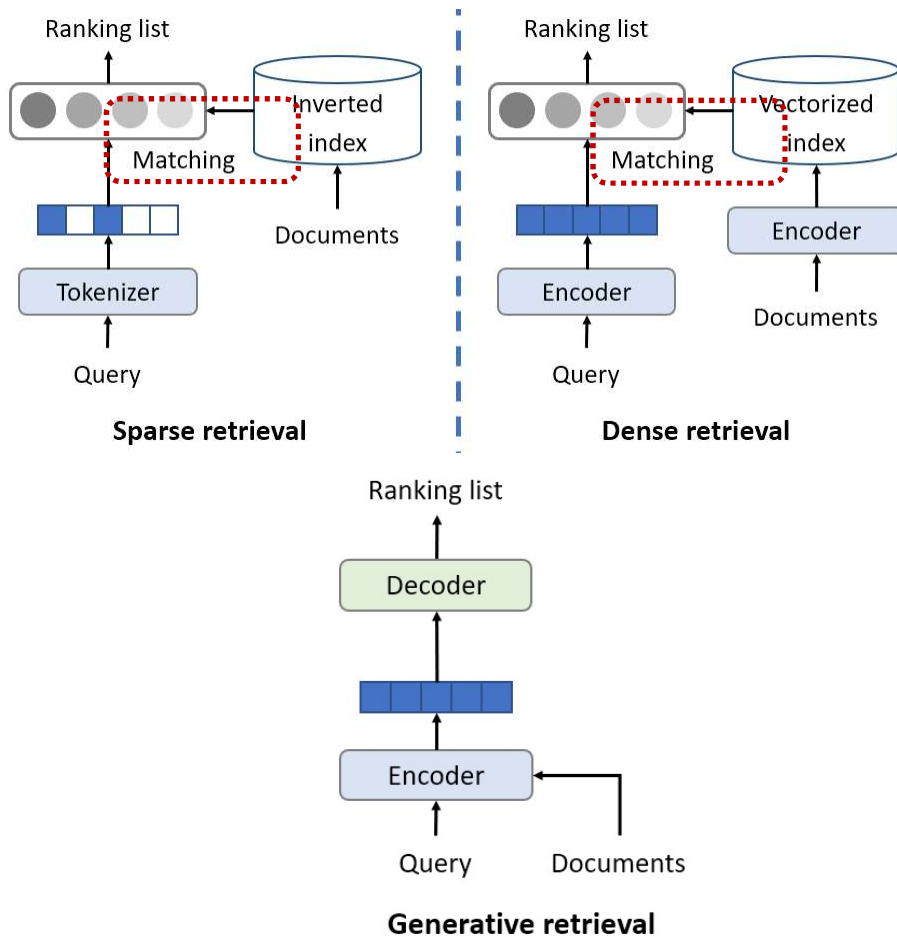


Generative Retrieval can parameterize the traditional **static index** with a pre-training model, which converts the document semantic mapping into a **dynamic and updatable process**.

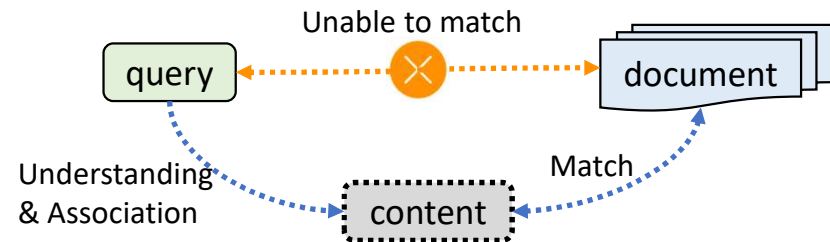
Sparse/Dense Retrieval vs. Generative Retrieval



Sparse/Dense Retrieval vs. Generative Retrieval

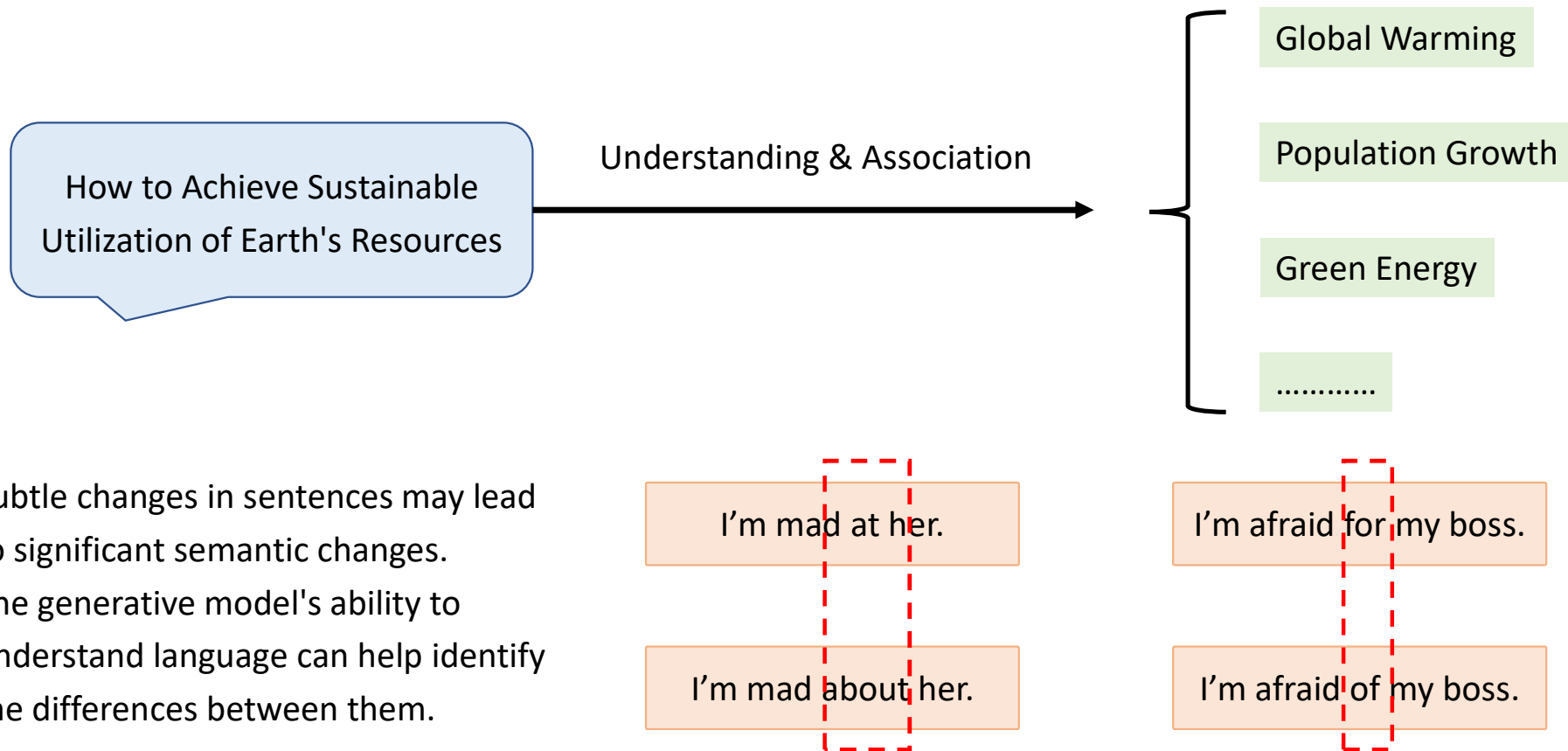


Traditional sparse retrieval and dense retrieval are both based on **matching** retrieval. These methods may lead to the lack of **fine-grained** semantic interaction.



Thanks to the powerful contextual modeling capabilities of LLMs, indexing systems based on big models are able to find documents that are related to queries in a **more associative & nuanced way**.

Sparse/Dense Retrieval vs. Generative Retrieval



- Subtle changes in sentences may lead to significant semantic changes.
- The generative model's ability to understand language can help identify the differences between them.

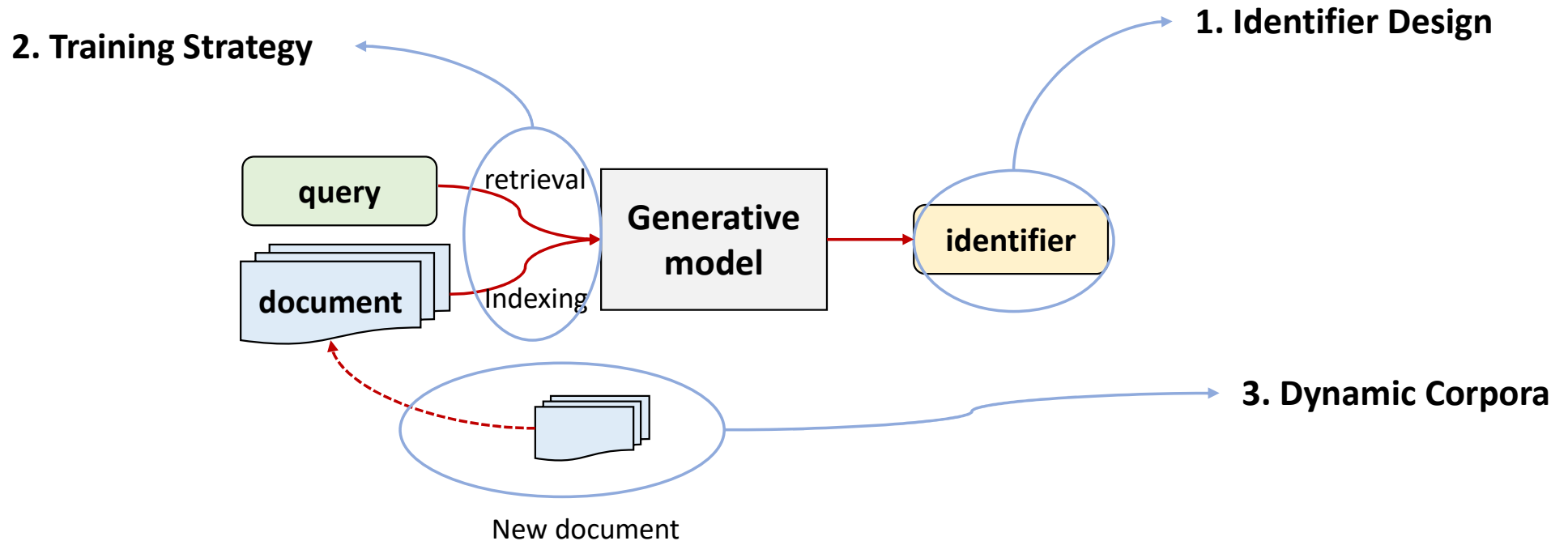


Outline

- Introduction to Retrieval Systems
- **Three Main Challenges of Generative Retrieval**
 - Identifier Design
 - Training Strategy
 - Dynamic Corpora
- Open Discussion

Three Main Challenges of Generative Retrieval

- Specifically, generative retrieval (GR) utilizes autoregressive language models to generate identifiers for relevant documents directly.

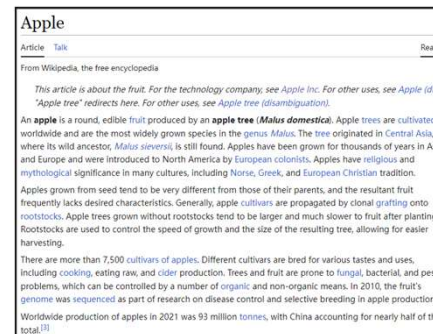


Identifier Design

Document Identifiers Design plays a crucial role in generative retrieval. And it is still an open problem on how to define the document identifiers.

- Different documents have short but different docids, and must **be distinctive enough** to represent a passage.
- Docids **capture the semantics** of their associated documents as much as possible.

Documents d



How to design identifier?



identifier

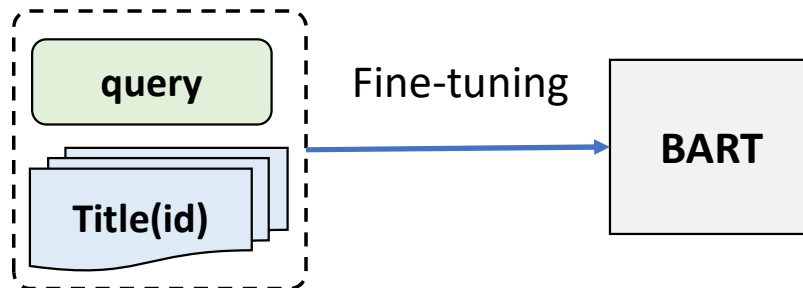
Identifier Design

- Lexical Type
 - Metadata-based approaches
 - Titles of documents L-1
 - URLs L-2
 - Document substring
 - N-grams L-3
 - Term-sets L-4
 - Synthetic identifiers
 - Generated pseudo-query L-5
- Numeric Type
 - Atomic ID N-1
 - Learned quantization (Semantic ID)
 - K-means cluster N-2
 - Tokenization learning method N-3

L-1 Titles of documents

GENRE

The first system that retrieves entities by generating their unique names, token-by-token in an autoregressive fashion.



- Retrieve an entity by generating the entity text itself.



Apply in passage-level retrieval

- Take the unique title contained in each document as an identifier.

- Using a standard seq2seq objective, i.e., maximizing the output sequence likelihood.

L-1 Titles of documents

- GENRE occupied **14 times less** memory than BLINK and **34 times less** memory than DPR.

$$\text{Memory} = \text{Model Param.} + \text{Index Size}$$

- In dense retrieval, the **index size** depends on **the number of dense vectors** corresponding to the documents.
- In generative retrieval, the **index size** depends on the size of a **prefix tree of the IDs**.



- Generative retrieval model uses less memory to store the entity index.

Dense
retrieval

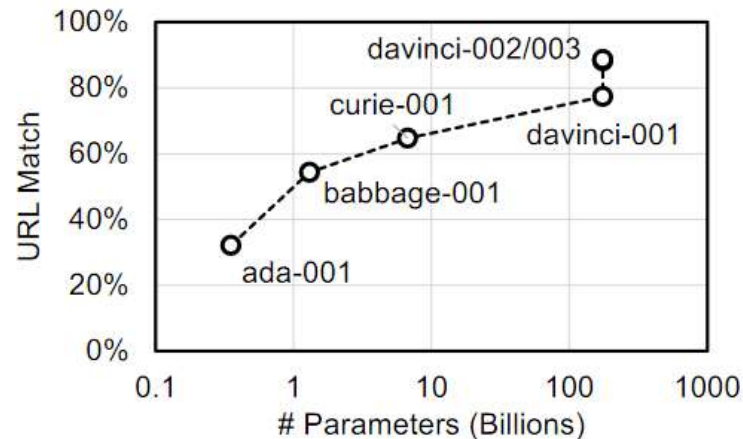
Model	Memory	Param.	Index
DPR	70.9GB	220M	15B
RAG	40.4GB	626M	15B
BLINK	30.1GB	680M	6B
GENRE	2.1GB	406M	17M

Generative
retrieval

L-2 URL as ID



View URL as the document identifier



- When providing a few Query-URL pairs as in-context demonstrations, LLMs can generate Web URLs where nearly **90%** of the corresponding documents contain correct answers to open-domain questions.

Intuitively, the URL of a document contains certain **semantic** information and can **uniquely** correspond to this document.

L-2 URL as ID

- TOME

Motivation

Manually or randomly constructed identifiers (e.g. Atomic ID) are not adequately captured in the pretraining stage of the generative PLM, thus limiting PLM's capabilities for generative prediction. This creates a **discrepancy between the pre-training and fine-tuning** phases.

`https://en.wikipedia.org/wiki/Natural_language_processing`



tokenize

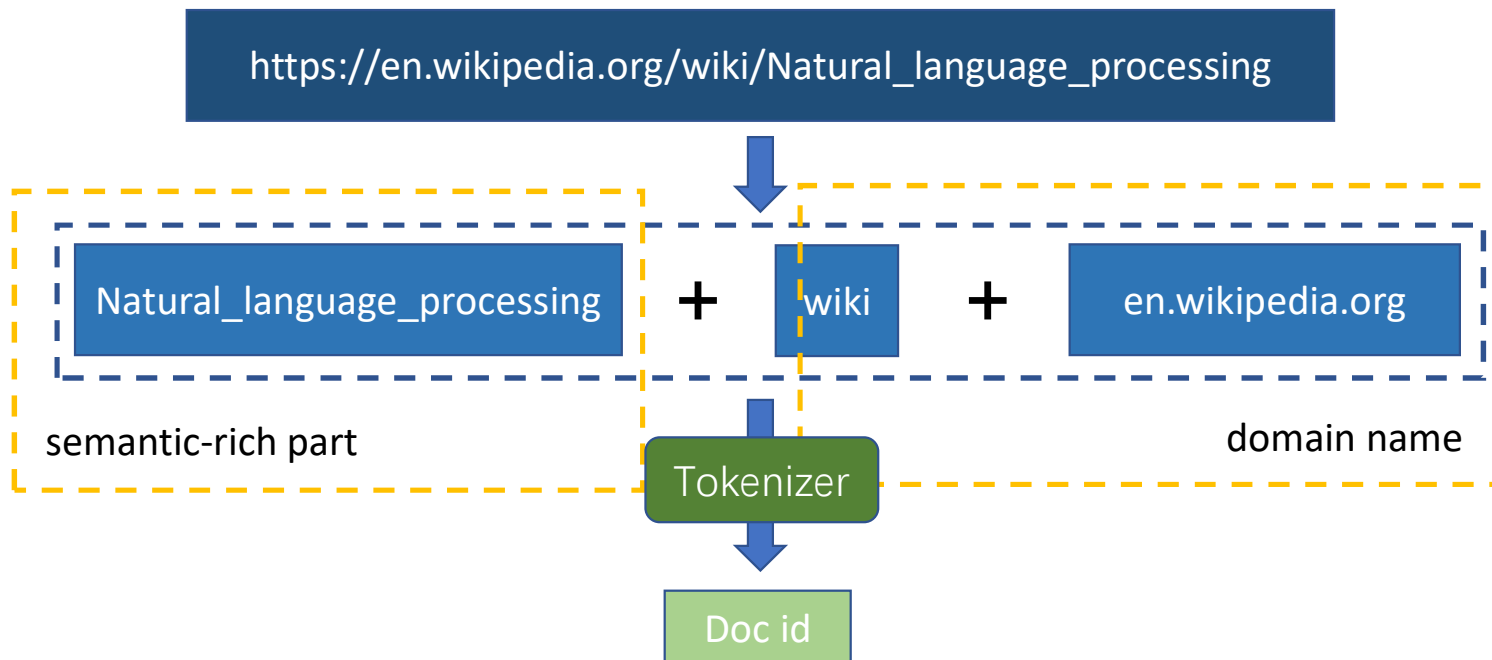
`"https", "://", "en", ".", "Wikipedia", ".", "org", "/", "wiki"`

- TOME solely utilizes **tokenized URLs** as the identifier, without any additional processing.
- A novel Two-stage Model-based retrieval approach based on T5 backbone. (I will introduce it in a later section)

L-2 URL as ID

- Ultron

$$\text{docid}_{\text{URL}} = \begin{cases} \text{reversed URL,} & \text{if title length} \leq L, \\ \text{title + domain,} & \text{otherwise.} \end{cases}$$



Unlike TOME, Ultron arranges each part of the URL (split by '/') in reverse order, so as to predict the semantic-rich part first, and then predict the domain name.

L-2 URL as ID

Model	Params	MS MARCO				Natural Questions (NQ)			
		R@1	R@5	R@10	MRR@10	R@1	R@5	R@10	MRR@10
Ultron-URL	248M (-)	0.2957 [†]	0.5643	0.6782	0.4002	0.3378 [‡]	0.5420 [‡]	0.6705	0.4251 [‡]
Ultron-PQ	257M (-)	0.3155 [†]	0.6398 [†]	0.7314	0.4535 [‡]	0.2564 [‡]	0.5309	0.6575	0.3712 [‡]
Ultron-Atomic	495M (↑)	0.3281 [‡]	0.6490 [‡]	0.7413 [‡]	0.4686 [‡]	0.2543 [‡]	0.5482 [‡]	0.6953 [‡]	0.3859 [‡]

- End-to-end retrievers with **atomic docids** achieve **better** results than those with **semantic docids**.
 - Random number as ids
 - Clustered numbers or natural language as ids
- More parameters make it easier to distinguish different documents.

- X However, atomic docids could lead to gigantic **parameters and memory burden**, especially when the number of documents increases.
- ✓ The URL docids using **sharable tokens** have the potential to alleviate these issues.

Identifier Design

In addition to using the meta data of document as id, there are also some methods that use **Document substring** as document id.

- Lexical Type
 - Metadata-based approaches
 - Titles of documents
 - URLs
 - Document substring
 - N-grams
 - Term-sets
 - Synthetic identifiers
 - Generated pseudo-query

L-3 N-grams as ID

Motivation



- ❑ The metadata such as title and URL are not always available.
- ❑ The metadata of the document cannot always completely cover all the information of the document.

SEAL

- Using all n-grams (substrings) in a passage as its possible identifiers

A kind of
“unsupervised”
data

SEAL VS Other identifier

Other

- ❑ The identifier of each document must be independent and different.

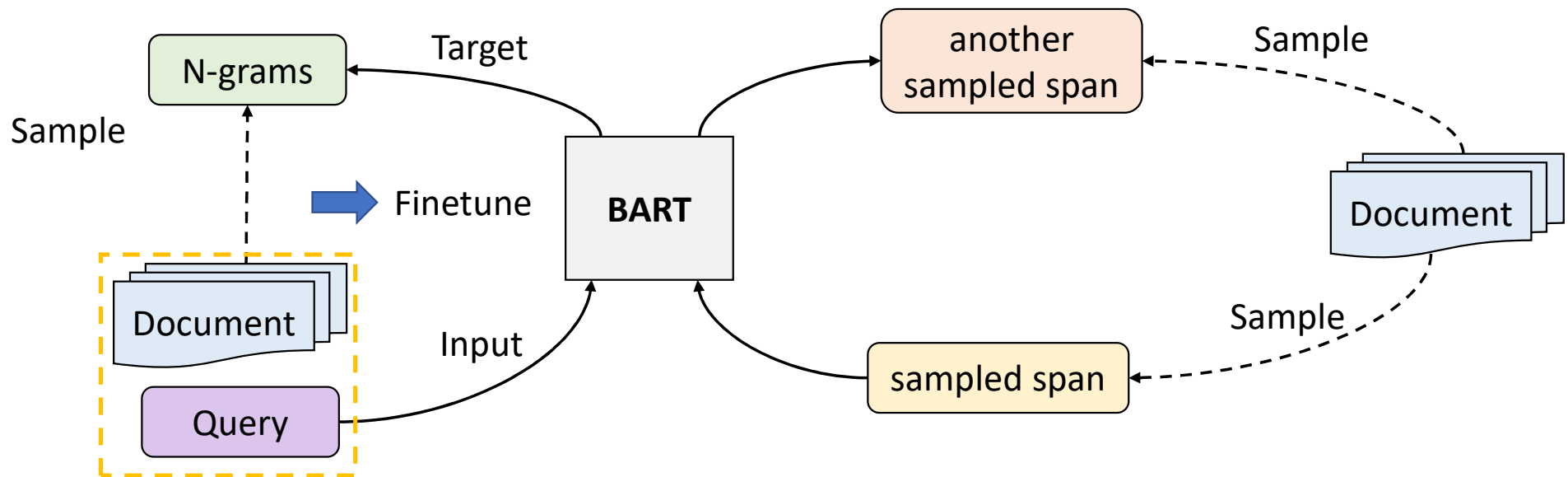
SEAL

- ❑ The identifiers do not necessarily occur in just one document.

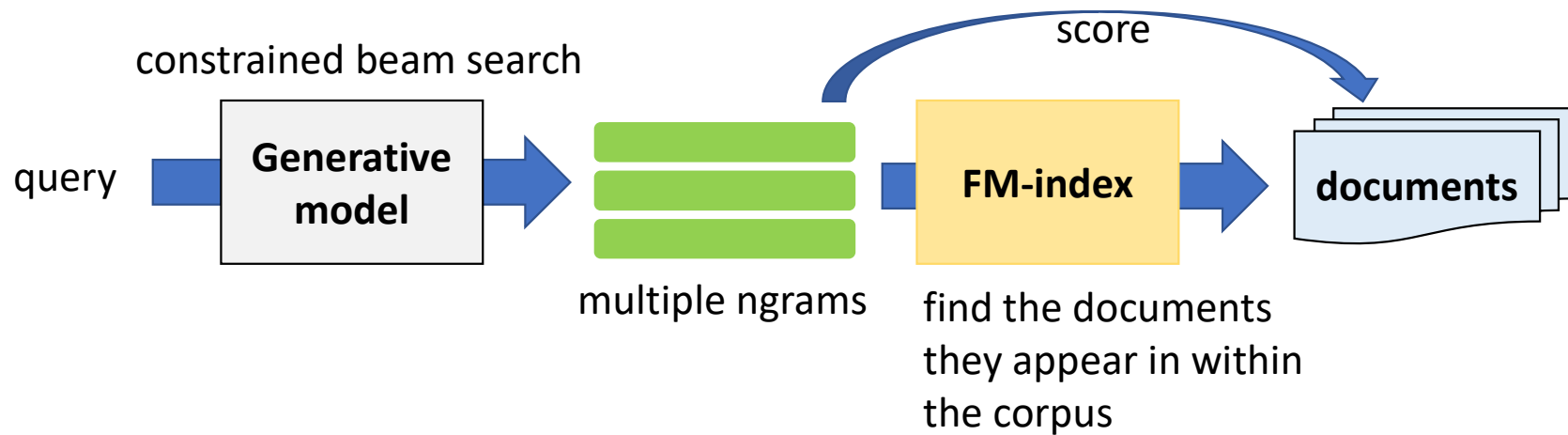
L-3 N-grams as ID

Training

- Finetune BART large to generate n-grams from the ground truth document.
- To **expose the model to more possible pieces of evidence**, they also add different “**unsupervised**” examples for each document in the retrieval corpus to the training set.



L-3 N-grams as ID



Core questions:

- How to generate multiple n-grams appeared in corpus?
- How to find documents based on the n-grams?
- How to score each document based on n-grams?

L-3 N-grams as ID

Core questions:

- How to generate multiple n-grams appeared in corpus?
- How to find documents based on the n-grams?
- How to score each document based on n-grams?



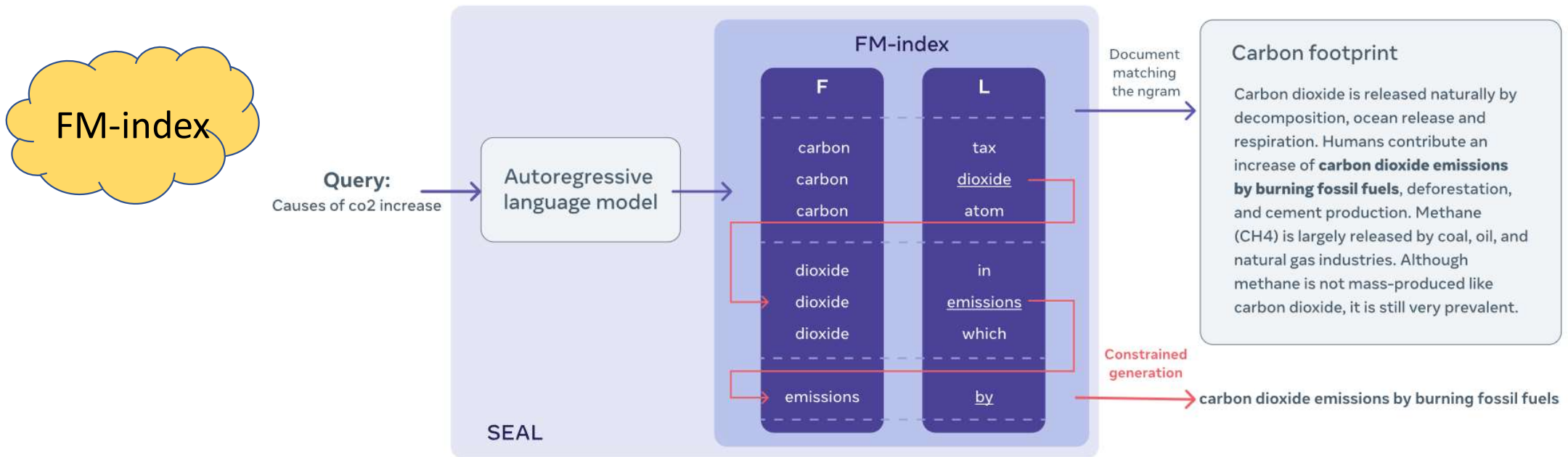
Use **FM-index** to **constrain decoding** to make sure that the generated n-grams are all valid.



For easy understanding

- FM-index could be regarded as a special prefix tree that supports search from any position.

L-3 N-grams as ID



The FM-index constraints the autoregressive generation (e.g., after “carbon”, the model is constrained to generate either “tax”, “dioxide” or “atom” in the example)

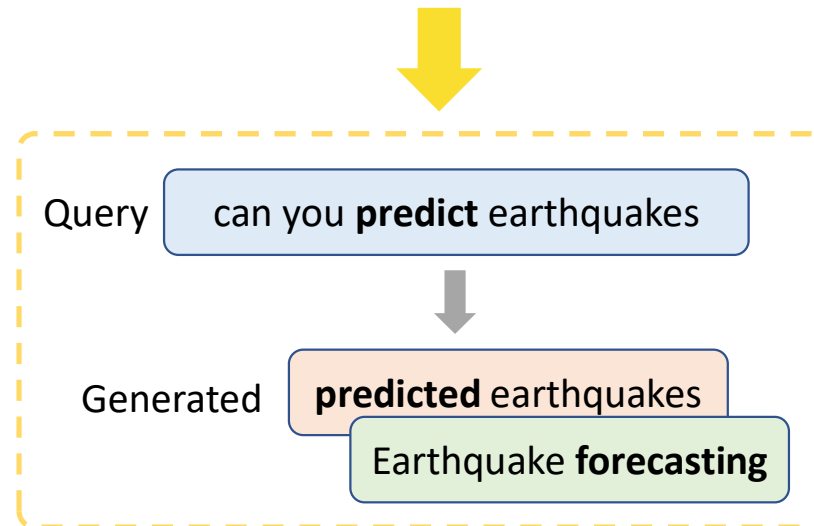
L-3 N-grams as ID

		①	②	
	score	# identifier	doc #1	doc #2
③	273.2	1 earthquakes can be predicted	Seismology @@ for precise earthquake predictions, including the VAN method. Most seismologists do not believe that a system to provide timely warnings for individual earthquakes has yet been developed, and many believe that such a system would be unlikely to give useful warning of impending seismic events. However, more general forecasts routinely predict seismic hazard. Such forecasts estimate the probability of an earthquake of a particular [...]	Earthquake prediction @@ reliably identified across significant spatial and temporal scales. While part of the scientific community hold that, taking into account non-seismic precursors and given enough resources to study them extensively, prediction might be possible, most scientists are pessimistic and some maintain that earthquake prediction is inherently impossible. Predictions are deemed significant if they can be shown to be successful beyond random chance.[...]
	272.7	75 Earthquake prediction @@		
	269.9	3 predicted earthquakes		
	229.7	11 Earthquake forecasting @@		
	217.2	2 prediction Earthquake		
	211.5	1 used to predict earthquakes		
	205.3	7 earthquakes. Earthquake		
	-77.0	9 Seismic metamaterial @@		
	-97.4	14 Seismic risk in Malta @@		
	-113.4	3 Quaternary (EP) @@		
-150.3	1 used to predict the locatio[...]			
-301.5	17 Precipice (Battlestar Gala[...]			

- ① Generate multiple n-grams.
- ② Resort FM-index to get documents which contain these n-grams.
- ③ Calculate the score of each n-gram and the score of each document and rank.

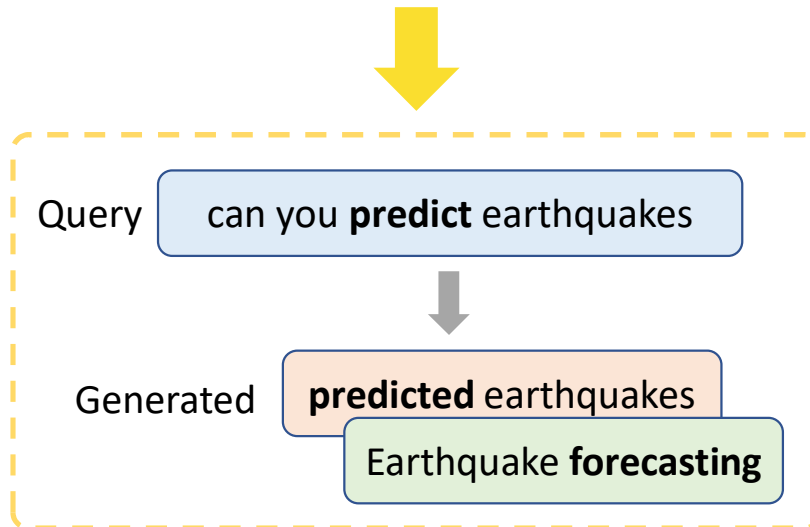
L-3 N-grams as ID

- In addition to being able to rephrase the query in ways that preserve its lexical material producing n-grams, SEAL can also explore more diverse regions of the output space, **overcoming the lexical mismatch problem.**



L-3 N-grams as ID

- In addition to being able to rephrase the query in ways that preserve its lexical material producing n-grams, SEAL can also explore more diverse regions of the output space, **overcoming the lexical mismatch problem.**



However, there is a problem:
for **long** documents, it is challenging
to enumerate **all possible n-grams.**

L-4 Term-sets as ID

Motivation



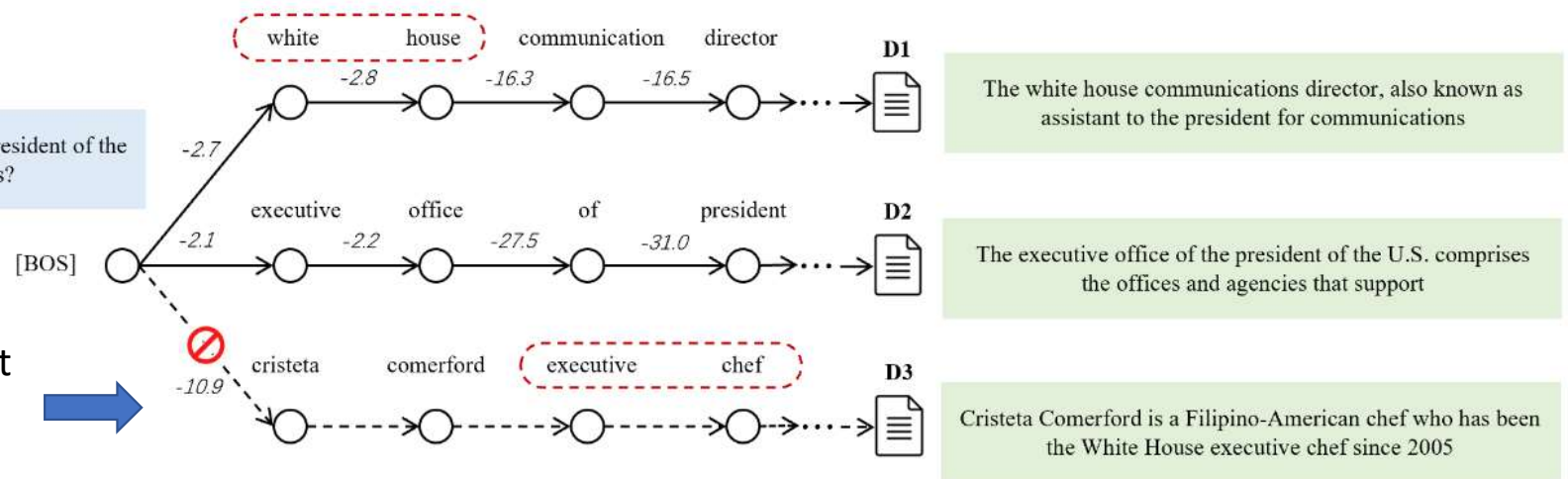
- Auto-regressive search engines **require the exact generation** of identifier for the targeted document. If incorrect predictions are made in any steps of the generation process, it will falsely produce the identifier of a different document.

AutoTSG

(A)

Beam Size=2

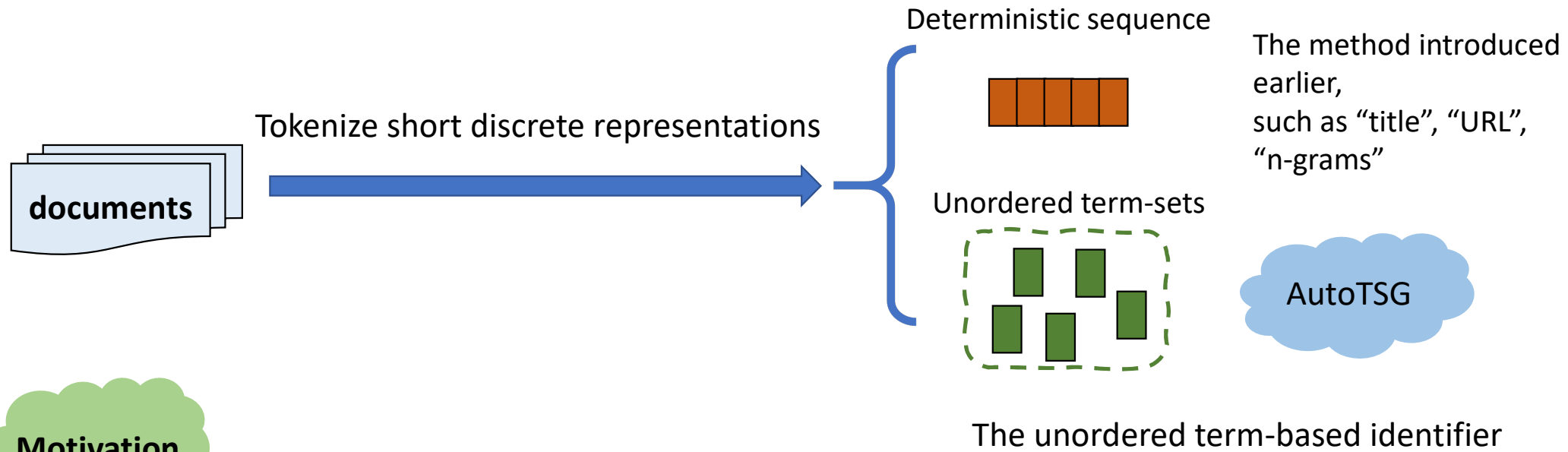
Q: who cooks for the president of the united states?



The correct result was ruled out in the previous decoding step.



L-4 Term-sets as ID



Motivation

- Relaxing the requirement of exact generation.
- Improving the generalization of retrieval generation quality.

L-4 Term-sets as ID

- The matching-oriented term selection



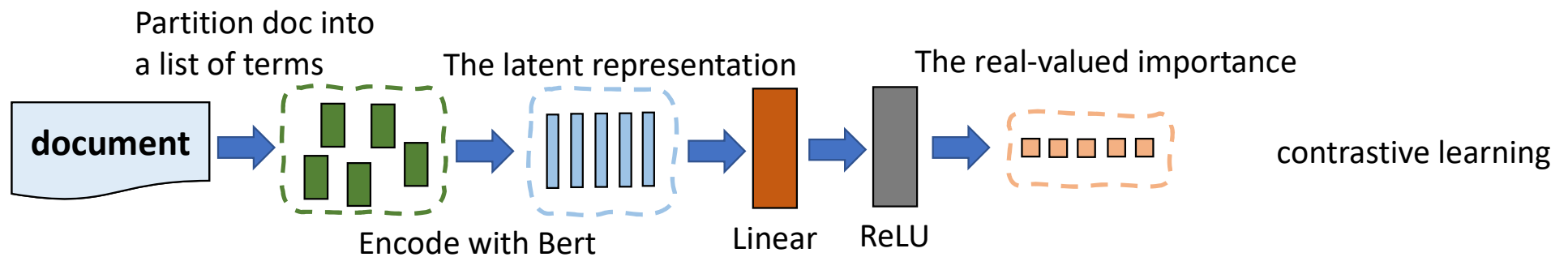
The selection of terms in a document identifier is performed based on the following principles.

- ❑ The number of selected terms N should be sufficiently large that all documents can be uniquely identified.
- ❑ To reduce the difficulty of prediction, the term selection needs to be concise as well.
- ❑ The selected terms must sufficiently capture the semantic information within the document.

L-4 Term-sets as ID

- The matching-oriented term selection

Representative terms are selected depending on their importance to the query-document matching.



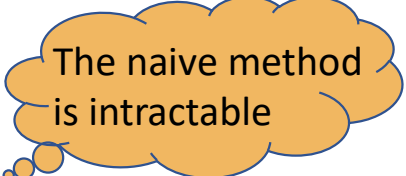
$$\mathcal{M}([t_1^D, \dots, t_L^D]) \xrightarrow{1.} [e_1^D, \dots, e_L^D] \xrightarrow{2.} [\sigma(W^T e_1^D), \dots, \sigma(W^T e_L^D)] \xrightarrow{3.} [w_1^D, \dots, w_L^D]$$

- Each document D is partitioned into a list of terms in the first place: $[t_1^D, \dots, t_L^D]$
- The encoding model $M(\cdot)$ is applied to transform each term t_i into its latent representation $e_i^D \in \mathbb{R}^{d \times 1}$
- Select the top-N terms as the identifier. $\mathcal{T}(D) \leftarrow \{t_i^D : w_i^D \in \text{top-}N(\{w_i^D\}_{i=1}^L)\}$



L-4 Term-sets as ID

- Constrained Greedy Search

- The generation likelihood is enumerated for all possible permutations of the document identifier ($N!$)
- The highest value is used as the measurement of relevance



The naive method is intractable

- 
- 
- ❑ **Optimality:** produce the document identifier of the highest generation likelihood.
 - ❑ **Validity:** The generated document identifier must be valid

L-4 Term-sets as ID

- Constrained Greedy Search

optimality

$$\{I_{\leq i}^*\}_K \leftarrow \operatorname{argtop}\text{-}K \left(\left\{ \prod_{j=1, \dots, i} \Pr(I_j \mid I_{< j}; Q; \Theta) \right\} \right).$$

- Greedily select the terms which give rise to the top-K generation likelihood until the current step.

validity

- Regularize the selection of I_i with the following set-difference based constraint:

$$1. I_i \notin \{I_1, \dots, I_{i-1}\} \wedge 2. \exists D : I_i \in \mathcal{T}(D) / \{I_1, \dots, I_{i-1}\}.$$

L-4 Term-sets as ID

Table 3: Analysis of retrieval quality w.r.t. **seen** and **unseen** documents on NQ320k.

Method	Seen (50%)		Unseen (50%)		Seen+Unseen (100%)	
	MRR@10	Recall@10	MRR@10	Recall@10	MRR@10	Recall@10
GENRE	0.763	0.869	0.138	0.187	0.448	0.558
DSI	0.713	0.802	0.011	0.040	0.360	0.428
Ultron	<u>0.782</u>	<u>0.891</u>	<u>0.300</u>	<u>0.383</u>	<u>0.471</u>	<u>0.570</u>
NCI	<u>0.751</u>	<u>0.842</u>	<u>0.050</u>	<u>0.159</u>	<u>0.393</u>	<u>0.459</u>
AutoTSG	0.809	0.900	0.466	0.654	0.552	0.700

Table 4: Analysis of retrieval quality w.r.t. **seen** and **unseen** documents on MS300k.

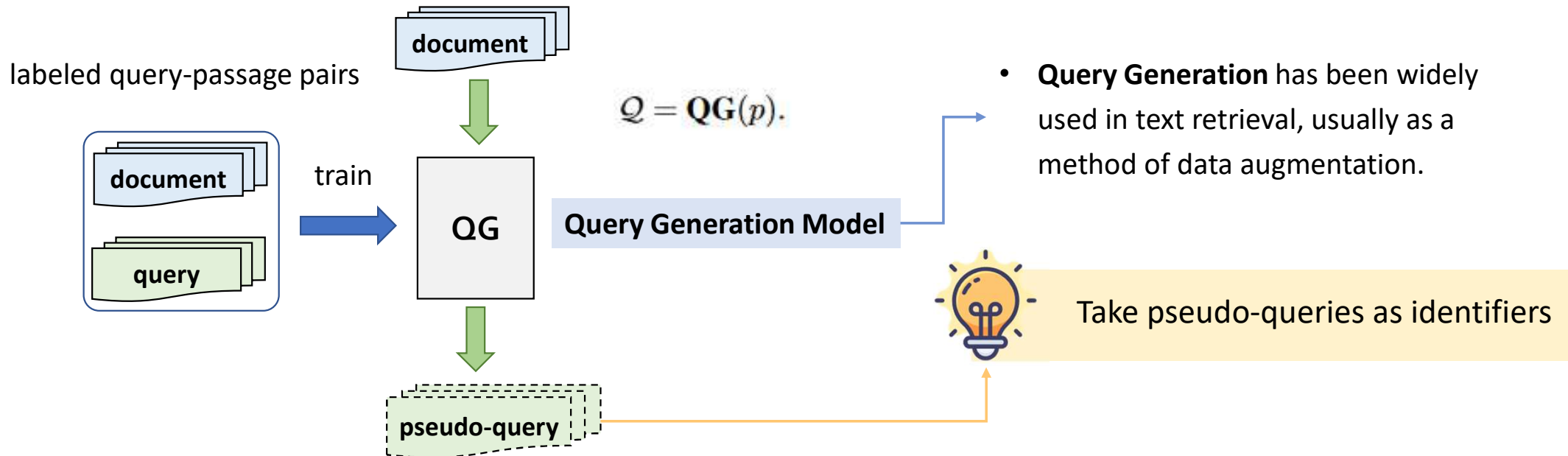
Method	Seen (50%)		Unseen (50%)		Seen+Unseen (100%)	
	MRR@10	Recall@10	MRR@10	Recall@10	MRR@10	Recall@10
GENRE	0.361	0.579	0.150	0.312	0.196	0.411
DSI	0.339	0.538	0.030	0.075	0.171	0.298
Ultron	<u>0.432</u>	<u>0.676</u>	<u>0.197</u>	<u>0.246</u>	<u>0.313</u>	<u>0.492</u>
NCI	<u>0.408</u>	<u>0.643</u>	<u>0.034</u>	<u>0.082</u>	<u>0.260</u>	<u>0.412</u>
AutoTSG	0.484	0.766	0.390	0.588	0.391	0.642

AutoTSG outperforms the baselines on the “seen” and “unseen” documents.

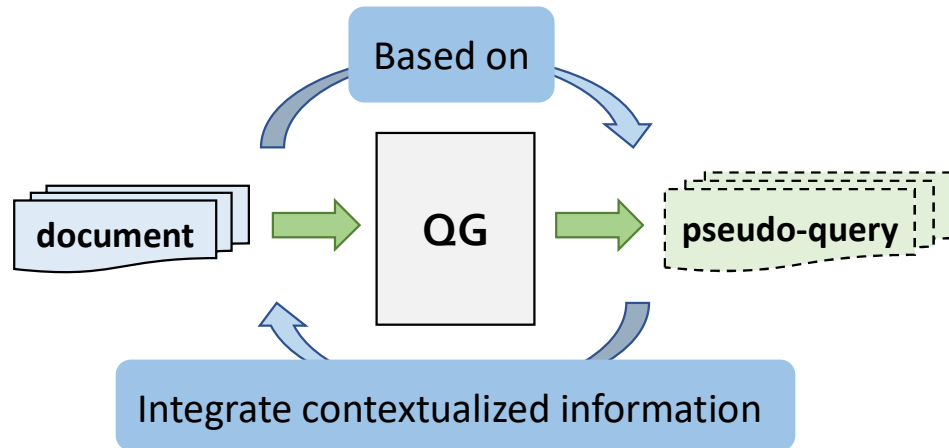
- Together with the flexibility to explore optimal identifier permutation, it becomes **more generalizable** when dealing with **unseen documents**.

L-5 Synthetic identifiers

In addition to using the existing data in documents as the identifiers mentioned above, there are also methods for **synthesizing identifiers**, such as **pseudo query**.



L-5 Synthetic identifiers



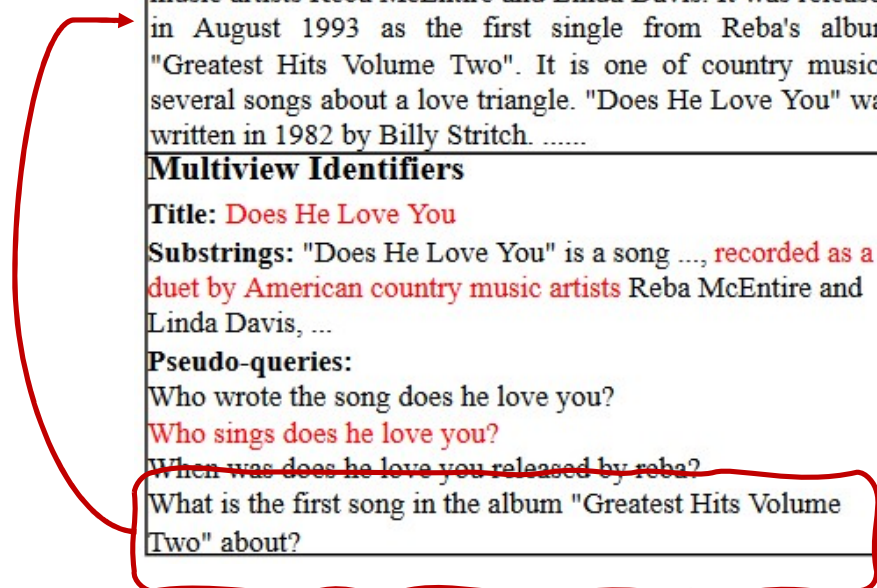
A query usually requires multiple segments of a document to answer.

Compared with title and substring, query can better **cover multiple segments** of the document, **containing more comprehensive semantics** in document.

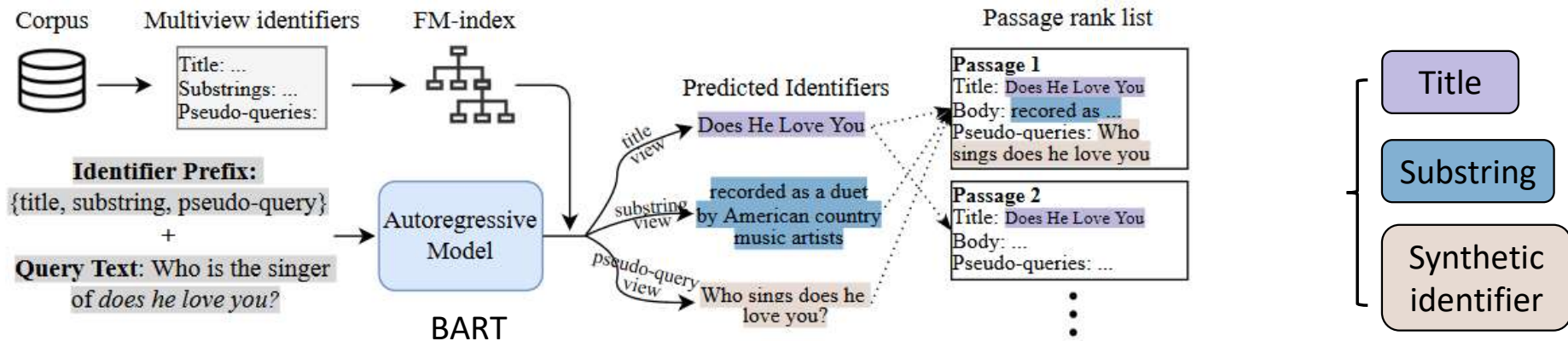
Query: Who is the singer of *does he love you*?

↑ Relevant

<p>Passage (https://en.wikipedia.org/wiki/Does_He_Love_You)</p> <p>"Does He Love You" is a song written by Sandy Knox and Billy Stritch, and recorded as a duet by American country music artists Reba McEntire and Linda Davis. It was released in August 1993 as the first single from Reba's album "Greatest Hits Volume Two". It is one of country music's several songs about a love triangle. "Does He Love You" was written in 1982 by Billy Stritch.</p> <p>Multiview Identifiers</p> <p>Title: Does He Love You</p> <p>Substrings: "Does He Love You" is a song ..., recorded as a duet by American country music artists Reba McEntire and Linda Davis, ...</p> <p>Pseudo-queries:</p> <p>Who wrote the song does he love you?</p> <p>Who sings does he love you?</p> <p>When was does he love you released by reba?</p> <p>What is the first song in the album "Greatest Hits Volume Two" about?</p>



Multiview identifiers



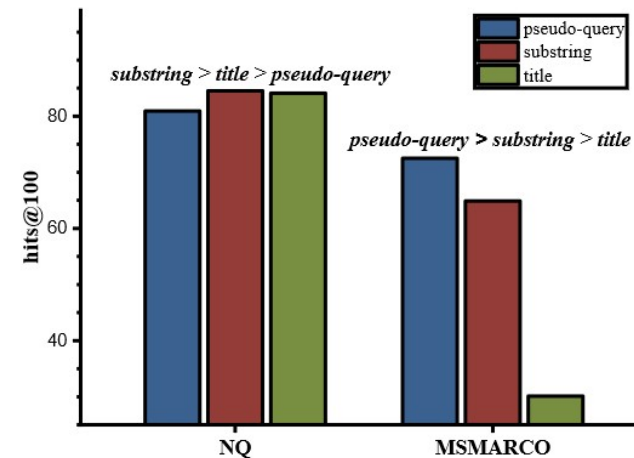
- Simultaneously consider multiview identifiers, including **synthetic identifiers, titles, and substrings**.
- For a passage p , select a subset I_p from the predicted identifiers.
- One identifier $i_p \in \{T_g, S_g, \text{and } Q_g\}$ is selected if i_p occurs at least once in the identifiers of passage p .

$$s(q, p) = \sum_{i_p \in \mathcal{I}_p} s_{i_p} \quad \text{where } s_{i_p} \text{ is the language model score of the identifier } i_p.$$

Multiview identifiers

- Experimental results show that it is better to consider three identifiers simultaneously.

Methods	Natural Questions		
	@5	@20	@100
only query	59.0	72.5	80.9
only substring	60.2	74.3	84.5
only title	60.4	74.9	84.1
w/o pseudo-query	63.4	77.2	86.1
w/o substring	63.1	77.0	85.0
w/o title	63.9	76.6	85.3
MINDER	65.8	78.3	86.7



- No matter which view of identifiers is removed, the performance significantly declines.
- The results verify the necessity to adopt multi-view identifiers simultaneously
- Different datasets tend to use different identifiers.
- Multiview identifiers can help improve generalization in different scenarios.



Identifier Design

- Lexical Type
 - Metadata-based approaches
 - Titles of documents L-1
 - URLs L-2
 - Document substring
 - N-grams L-3
 - Term-sets L-4
 - Synthetic identifiers
 - Generated pseudo-query L-5
- Numeric Type
 - Atomic ID N-1
 - Learned quantization (Semantic ID)
 - K-means cluster N-2
 - Tokenization learning method N-3

Atomic ID

N

A naive way to resort numeric identifier is to assign each an arbitrary (and possibly random) unique integer identifier.

- ✓ Results have proved that this way can achieve remarkable performance.
- X But this random identifier does not contain the semantic information of the document.
- X As the corpus size increases, the vocabulary used in decoding also expands, leading to an increase in model parameters.

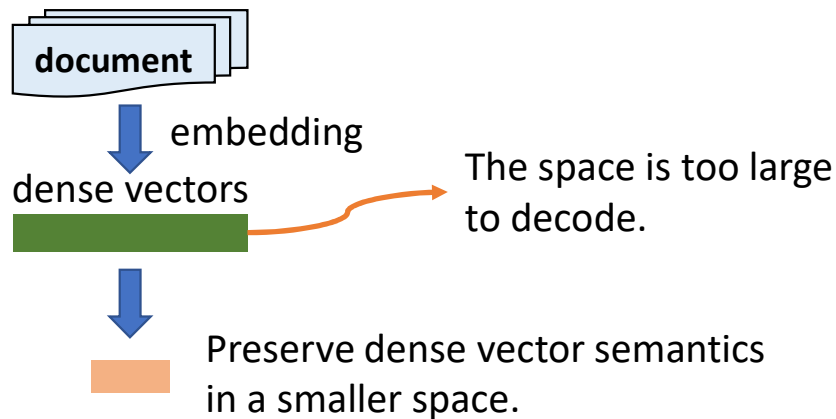
More parameters make it easier to distinguish different documents.

Semantically Structured Identifiers

Motivation



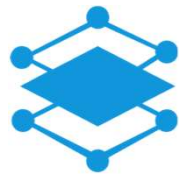
Dense vectors can be used as identifiers, but **the space of dense vectors is too large to decode**. This promotes us to look for a way to preserve dense vector semantics in a smaller space. Imbuing the docid space with **semantic structure** can lead to better indexing and retrieval capabilities.



Semantically Structured Identifiers need to satisfy the following **properties**:

- ❑ They should capture some information about the semantics of its associated document.
- ❑ They should be structured in a way that the search space is effectively reduced after each decoding step.

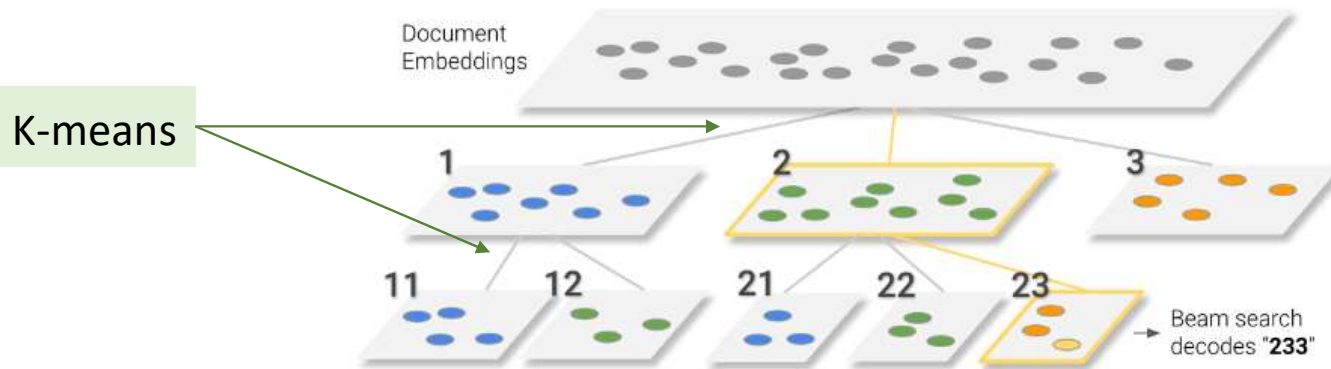
Differentiable Search Index (DSI)



Hierarchical clustering

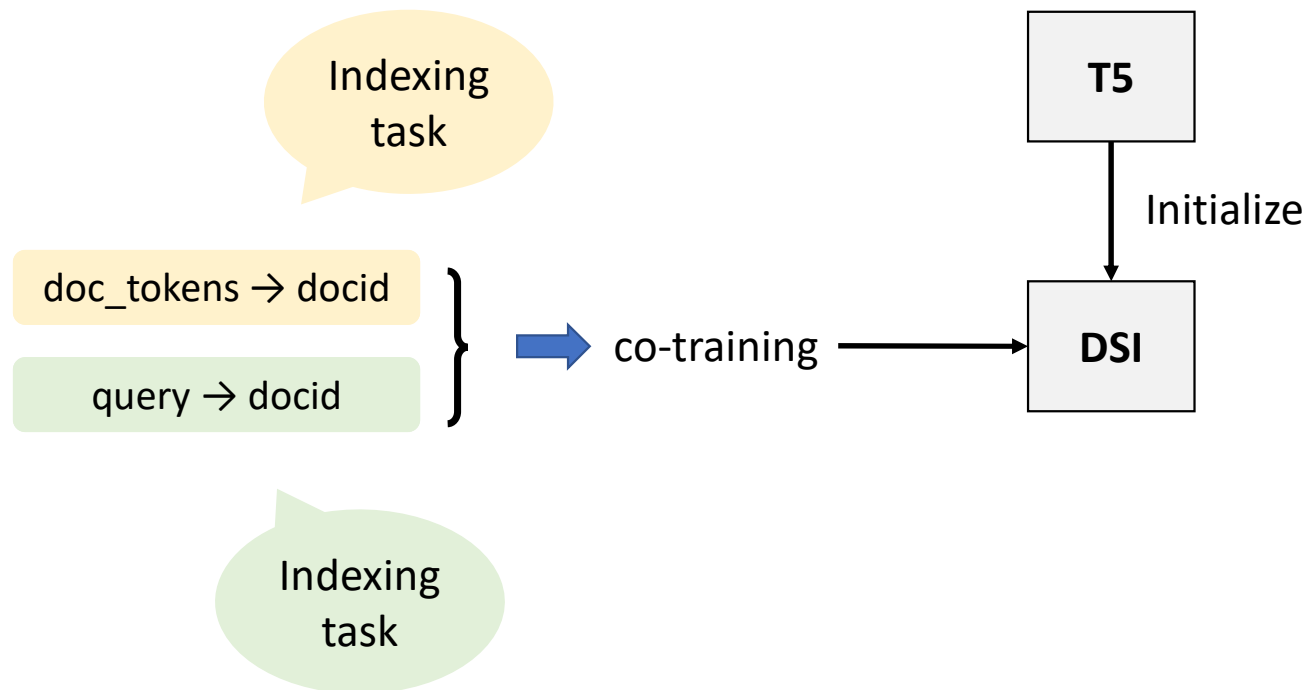
Imbuing the target space with semantic structure can facilitate greater ease of optimization and additional unsupervised representation learning methods as external knowledge.

A fully unsupervised pre-processing step



Differentiable Search Index (DSI)

- Frame co-training tasks in similar fashion to T5-style co-training



DSI result

Model	Size	Params	Method	NQ10K		NQ100K		NQ320K	
				Hits@1	Hits@10	Hits@1	Hits@10	Hits@1	Hits@10
BM25	-	-	-	12.4	33.5	20.9	46.4	11.6	34.4
T5	Base	220M	Dual Encoder	16.2	48.6	18.7	55.2	20.5	58.3
T5	Large	800M	Dual Encoder	18.8	55.7	22.3	60.5	22.4	63.3
T5	XL	3B	Dual Encoder	20.8	59.6	23.3	63.2	23.9	65.8
T5	XXL	11B	Dual Encoder	22.1	61.6	24.1	64.5	24.3	67.3
DSI	Base	250M	Atomic Docid	13.0	38.4	23.8	58.6	20.7	40.9
DSI	Large	800M	Atomic Docid	31.3	59.4	17.1	52.3	11.6	37.6
DSI	XL	3B	Atomic Docid	40.1	76.9	19.0	55.3	28.1	61.9
DSI	XXL	11B	Atomic Docid	39.4	77.0	25.3	67.9	24.0	55.1
DSI	Base	250M	Naive String Docid	28.1	48.0	18.7	44.6	6.7	21.0
DSI	Large	800M	Naive String Docid	34.7	60.5	21.2	50.7	13.3	33.6
DSI	XL	3B	Naive String Docid	44.7	66.4	24.0	55.1	16.7	58.1
DSI	XXL	11B	Naive String Docid	46.7	77.9	27.5	62.4	23.8	55.9
DSI	Base	250M	Semantic String Docid	33.9	57.3	19.0	44.9	27.4	56.6
DSI	Large	800M	Semantic String Docid	37.5	65.1	20.4	50.2	35.6	62.6
DSI	XL	3B	Semantic String Docid	41.9	67.1	22.4	52.2	39.1	66.8
DSI	XXL	11B	Semantic String Docid	48.5	72.1	26.9	59.5	40.4	70.3

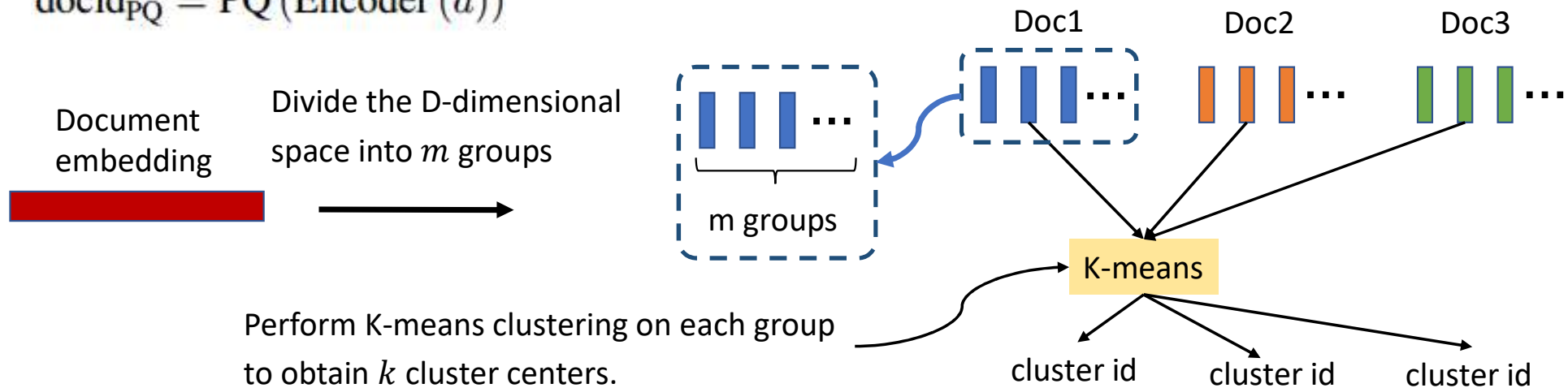
- The effect of unstructured atomic identifiers is unstable and they may face difficulties in optimization. (**instability and high variance**)
- Imbuing the target space with semantic structure can **facilitate greater ease of optimization** and additional unsupervised representation learning methods as external knowledge.

Product Quantization

- Ultron attempted another method of hierarchical clustering to obtain docids.

product quantization (PQ) - a much more powerful and flexible way to **quantize vectors**. Essentially, it's a method of data compression representation.

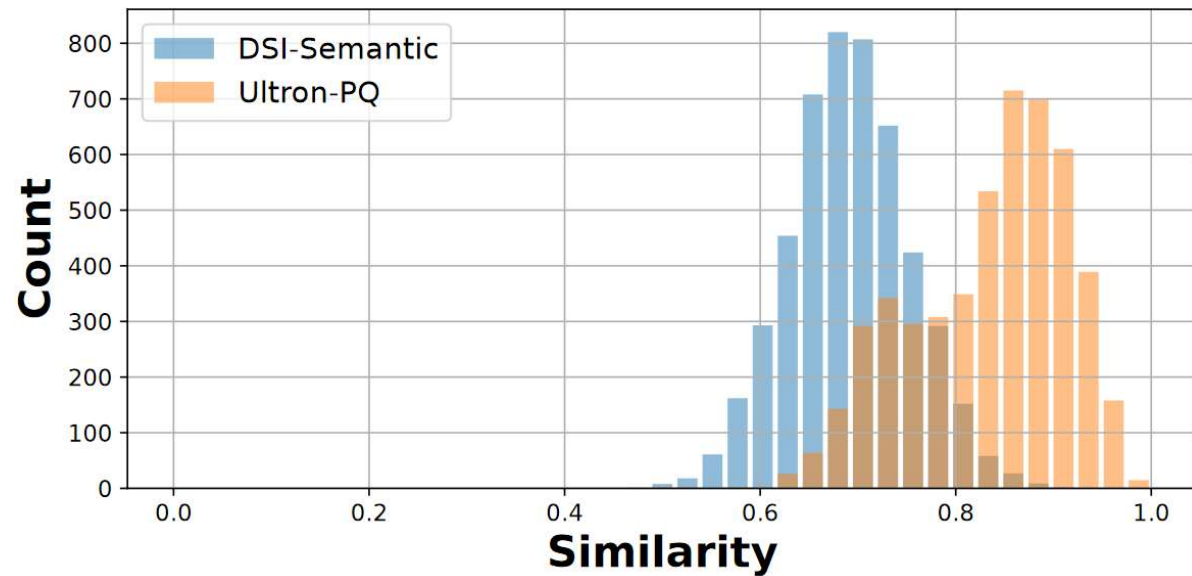
$$\text{docid}_{\text{PQ}} = \text{PQ}(\text{Encoder}(d))$$



Product Quantization



Compared with DSI directly using **k-means** clustering, whether using **PQ** can better imply semantics?



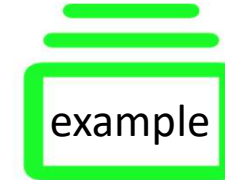
Distribution of similarity between the embeddings of every two documents with the same prefixes (length = 2)

Prefix-aware weight-adaptive decoder (PAWA)



Although the semantic ID designed using the clustering method has hierarchical semantic information, this **hierarchical structure** is not fully leveraged in the decoding stage.

Why is hierarchical structure important?



- The meanings of the same token appearing at different places of the same identifier are different, as they correspond to different clusters in the hierarchical tree structure.
- The same token in the same position may have different semantics with different prefixes.

The “ 5_2 ” and “ 5_3 ” of the same identifier “ $3_1 5_2 5_3$ ” correspond to different semantic meanings.

In identifiers “ $1_1 1_2 5_3$ ” and “ $2_1 4_2 5_3$ ”, the same token “ 5_3 ” has different semantics in two different identifiers.

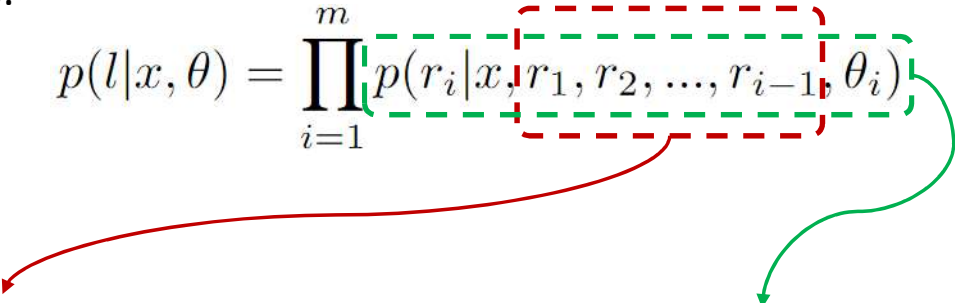
Prefix-aware weight-adaptive decoder (PAWA)

Motivation

Fully leverage the hierarchical structure of semantic ID in the decoding stage.

r_i is the i -th token in the current identifier
 θ_i is the parameter for the i -th step

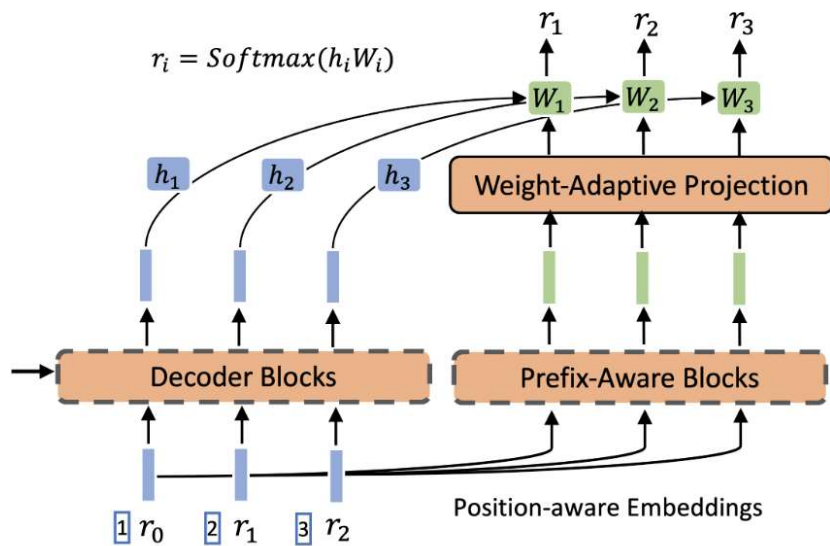
Given an input query x , the probability of generating a document identifier can be written as:

$$p(l|x, \theta) = \prod_{i=1}^m p(r_i|x, r_1, r_2, \dots, r_{i-1}, \theta_i)$$




How to make full use of r_1, r_2, \dots, r_{i-1} when calculating $p(r_i|x, r_1, r_2, \dots, r_{i-1}, \theta_i)$

Prefix-aware weight-adaptive decoder (PAWA)



Instead of using the same projection weight W in the linear classification layer, PAWA employ the prefix-aware adaptive weights for each token classifier.

Standard decoder

$$h_i = \text{TransformerDecoder}(x, h_1, h_2, \dots, h_{i-1}; \theta_i),$$

$$p(r_i | x, r_1, r_2, \dots, r_{i-1}, \theta_i) = \text{Softmax}(h_i W). \quad W \in \mathbb{R}^{d \times v}$$

Prefix-Aware Weight-Adaptor (PAWA) decoder

$$h_i = \text{TransformerDecoder}(x, h_1, h_2, \dots, h_{i-1}; \theta_i),$$

$$p(r_i | x, r_1, r_2, \dots, r_{i-1}, \theta_i) = \text{Softmax}(h_i W_{ada}^i).$$

$$W_{ada}^i = \text{AdaptiveDecoder}(e; r_1, r_2, \dots, r_{i-1}) W_i$$

Stacks transformer decoding layers

Prefix-aware weight-adaptive decoder (PAWA)

Table 3: Ablation Study on NQ320k and TriviaQA retrieval task.

Method	NQ320k				TriviaQA			
	Recall@1	Recall@10	Recall@100	MRR@100	Recall@5	Recall@20	Recall@100	R-Precision
Neural Corpus Indexer (Base)	65.86	85.20	92.42	73.12	90.49	94.45	96.94	73.90
w/o DocT5Query	60.23	80.20	90.92	67.89	84.56	90.94	95.32	63.50
w/o document as query	62.49	81.21	88.85	69.41	85.34	91.10	94.66	67.48
w/o PAWA decoder	63.36	83.06	91.47	70.56	88.75	93.56	96.18	71.81
w/o semantic id	62.75	83.88	91.01	70.43	88.91	93.07	95.80	72.57
w/o regularization	65.07	82.91	90.65	71.80	89.01	93.63	96.16	71.59
w/o constrained beam search	65.65	84.89	92.23	72.79	89.58	93.97	96.61	72.51

The prefix-aware weight-adaptive decoder has a noticeable influence on the performance, which indicates that, instead of borrowing the vanilla transformer decoder, it is necessary to design a tailored decoder architecture for the task of semantic identifier generation.

Tokenization learning method

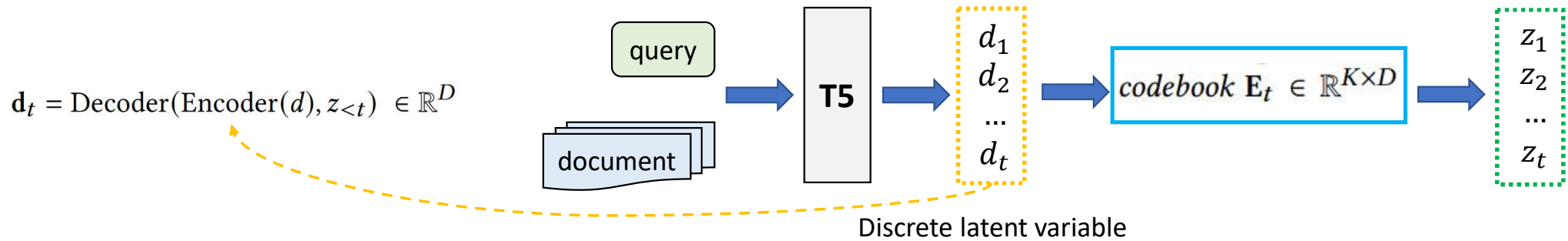
- GENRET

- A novel **tokenization learning** method based on discrete auto-encoding

These methods often fail to capture the complete semantics of a document.



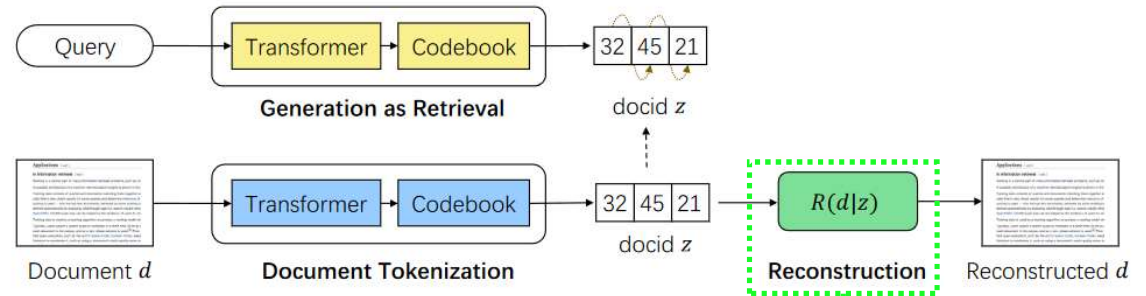
Previous work has treated identifier generation as a fully unsupervised **pre-processing** step. How to integrate and automatically learn semantic identifiers in a **fully end-to-end manner**?



Tokenization learning method

Model optimization

The learning objective for document tokenization consisting of three loss functions



- Retrieval loss

$$\mathcal{L}_{\text{Ret}} = -\log \frac{\exp(\mathbf{q}_T \cdot \mathbf{d}_T)}{\sum_{d^- \sim B} \exp(\mathbf{q}_T \cdot \mathbf{d}^-_T)} - \sum_{t=1}^T \log P(z_t | z_{<t}, q)$$

- ranking-oriented loss
- cross-entropy loss

- Commitment loss

$$\mathcal{L}_{\text{Com}} = -\sum_{t=1}^T \log Q(z_t | z_{<t}, d)$$

- To make sure the predicted docid commits to an embedding and to avoid models forgetting previous docid $z_{<t}$

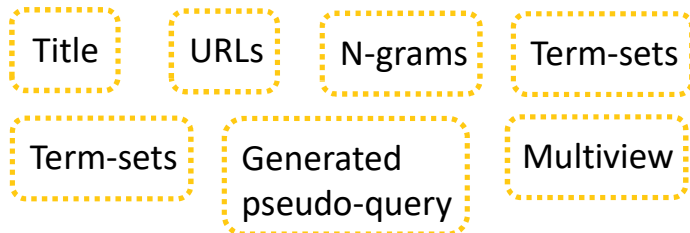
- Reconstruction loss

$$\mathcal{L}_{\text{Rec}} = -\log R(d | \hat{\mathbf{z}}_{\leq T}) \rightarrow R(d | \mathbf{z}) = \prod_{t=1}^M \frac{\exp(\mathbf{z}_t \cdot \text{sg}(\mathbf{d}_t^T))}{\sum_{d^* \in S(\mathbf{z}_{<t})} \exp(\mathbf{z}_t \cdot \text{sg}(\mathbf{d}^{*T}_t))}$$

- The relevance score between the input docid \mathbf{z} and the target document d

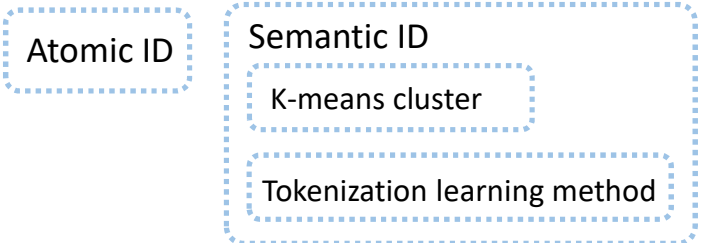
Identifier Design

Lexical Type



Section summary

Numeric Type



- ✓ Compared to non-linguistic IDs, Lexical identifier typically contain **more meaningful tokens** that widely exist in real-world corpora, making them more suitable to modeling and prediction using generative PLMs.
- X Using meta-data like title, URL, substring, etc. as IDs maybe has **low relevance** to the content of the documents.
- X **Some meta-data often does not exist**, which limits the generalizability of the task.

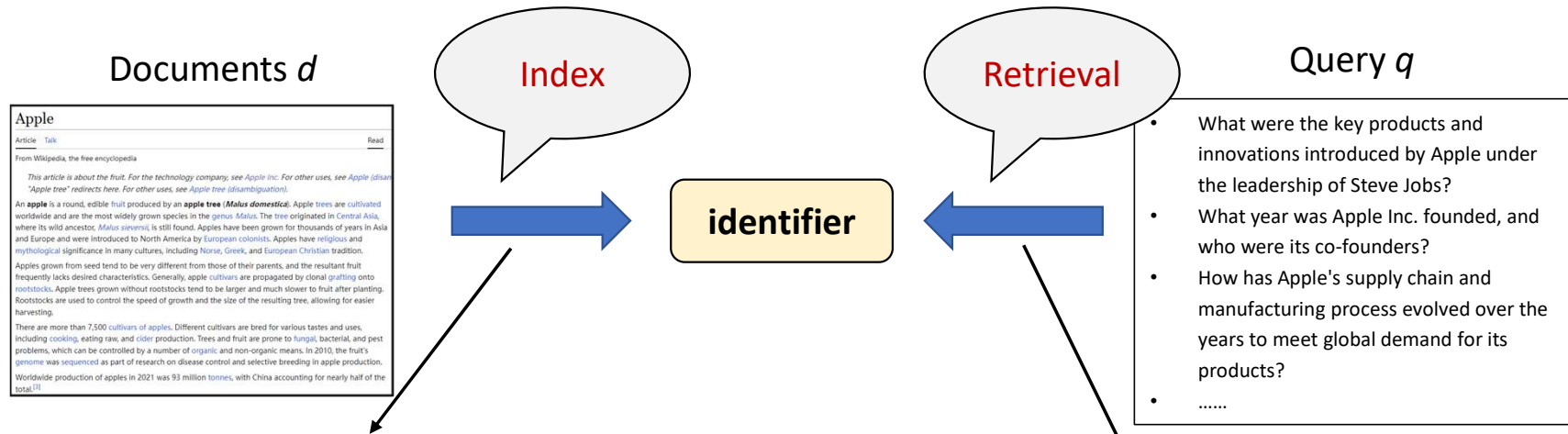
- ✓ The construction of semantic ID can help **better capture the complete semantics** of a document.
- X These IDs are often randomly generated or manually constructed, which may **not exist in real world corpora**. However, the generative PLM is pre-trained based on large-scale text corpora, leading to a **discrepancy between pre-training and fine-tuning**.
- X Numeric identifiers **require extra steps to memorize the mapping from passages to IDs**.



Outline

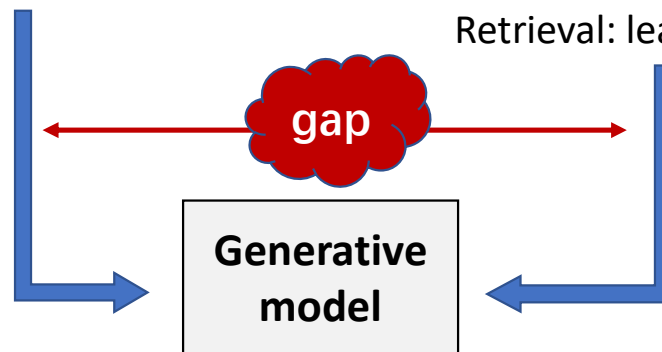
- Introduction to Retrieval Systems
- Three Main Challenges of Generative Retrieval
 - Identifier Design
 - **Training Strategy**
 - Dynamic Corpora
- Open Discussion

Training Strategy



Index: memorize information about each document

Retrieval: learn how to effectively retrieve from the index

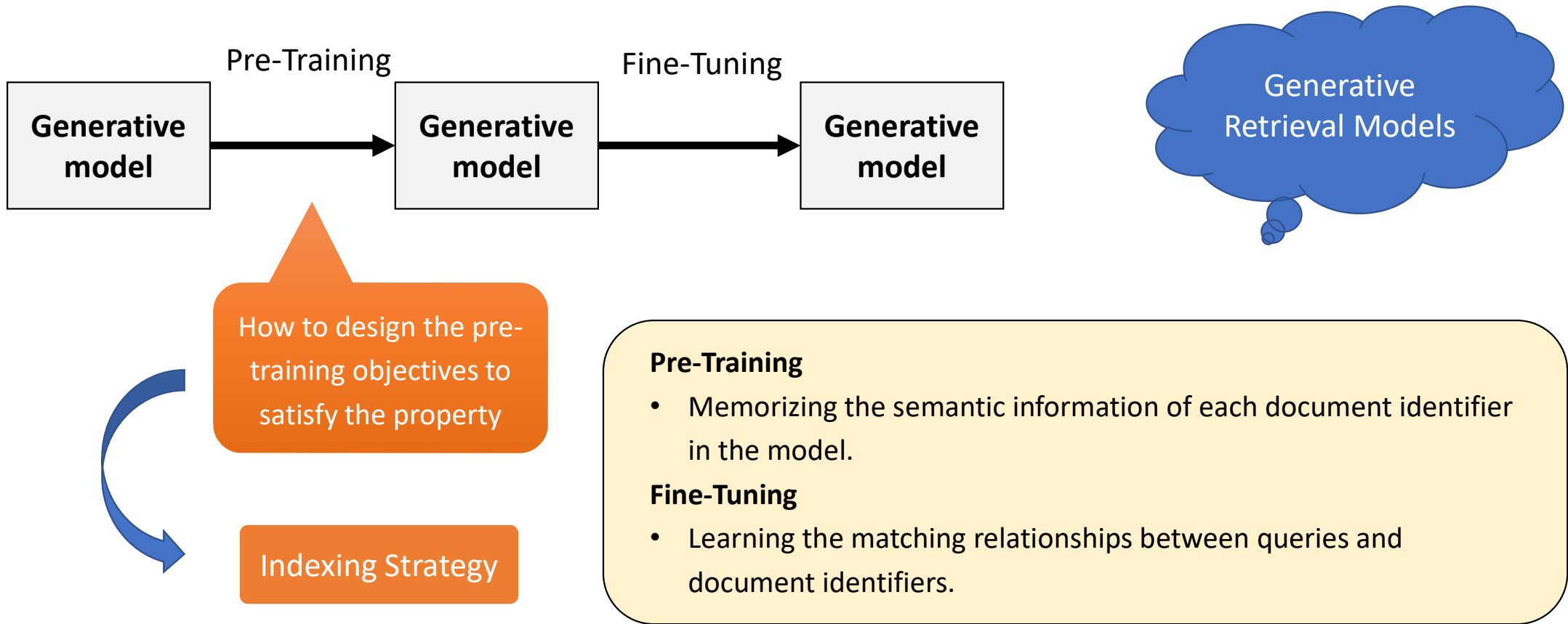




Training Strategy

- General Pre-train S-1
- Data Augmentation S-2
 - Query Generation Model
- Multi-stage Generation Architecture S-3
 - Two-stage generation architecture (TOME)
- Learning to Rank S-4

S-1 General Pre-train



Indexing Strategy

Indexing Strategy

How to learn associations between documents and their identifiers.



Best performance

☐ Inputs2Target $\text{doc_tokens} \rightarrow \text{docid}$

Bind the docids to the document tokens in a straightforward inputs-to-targets fashion.

- ✓ The identifier is the denoising target, which puts it in closer proximity to the loss function.

☐ Targets2Inputs $\text{docid} \rightarrow \text{doc_tokens}$

Intuitively, this is equivalent to training an autoregressive language model that is conditioned on the docid.

☐ Bidirectional

$\left\{ \begin{array}{l} \text{doc_tokens} \rightarrow \text{docid} \\ \text{docid} \rightarrow \text{doc_tokens} \end{array} \right. \rightarrow \text{co-training}$

☐ Span Corruption

$\underbrace{\text{docid} \text{ doc_tokens}}_{\text{concatenate}} \rightarrow \text{Randomly masked as spans in the span corruption objective.}$

Indexing Strategy

❑ Inputs2Target

doc_tokens → docid



❑ Bidirectional

doc_tokens → docid

docid → doc_tokens

Hits@1 (13.5 vs 13.2)

Performs the best

Not working

❑ Targets2Inputs

docid → doc_tokens

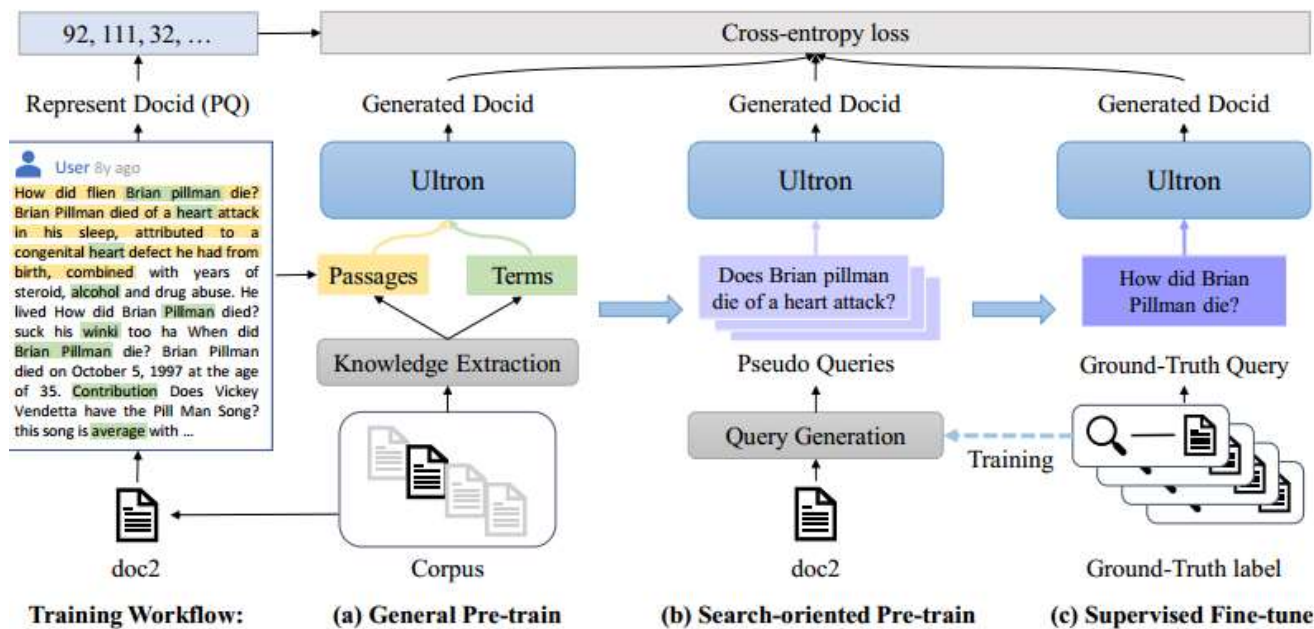
❑ Span Corruption

docid doc_tokens

Yield no meaningful results (0% accuracy)

There can be huge variance across indexing strategies whereby some strategies work reasonably well and some completely do not work at all.

Three-stage training workflow



Ultron is a three-stage training workflow to capture more knowledge contained in the corpus and associations between queries and docids.

Three-stage training workflow

- General Pre-train
- Search-oriented Pre-train
- Supervised Fine-tune

Three-stage training workflow

- Verify the effects of each training stage on the final results

Model	MS MARCO		Natural Questions	
	MRR@10	R@10	MRR@10	R@10
Ultron-URL	0.4002	0.6782	0.4251	0.6705
w/o general pretrain	0.3856	0.6321	0.3587	0.6608
w/o search-oriented	0.3341	0.5211	0.3071	0.6147
w/o finetune	0.3477	0.5693	0.3504	0.6405
Ultron-PQ	0.4535	0.7314	0.3712	0.6575
w/o general pretrain	0.4099	0.6968	0.3328	0.6327
w/o search-oriented	0.3445	0.5730	0.2427	0.5220
w/o finetune	0.4176	0.7023	0.3522	0.6386

- ❑ The removal of each training stage will decrease the results on all evaluation metrics.
- ❑ General pre-training stage is dedicated to gaining the document knowledge of each docid.
- ❑ The pseudo query-docid pairs significantly enhance the model's performance.
- ❑ The only supervised stage (ground-truth query-id) is necessary.



Training Strategy

- General Pre-train S-1
- Data Augmentation S-2
 - Query Generation Model
- Multi-stage Generation Architecture S-3
 - Two-stage generation architecture (TOME)
- Learning to Rank S-4

DSI + Query Generation



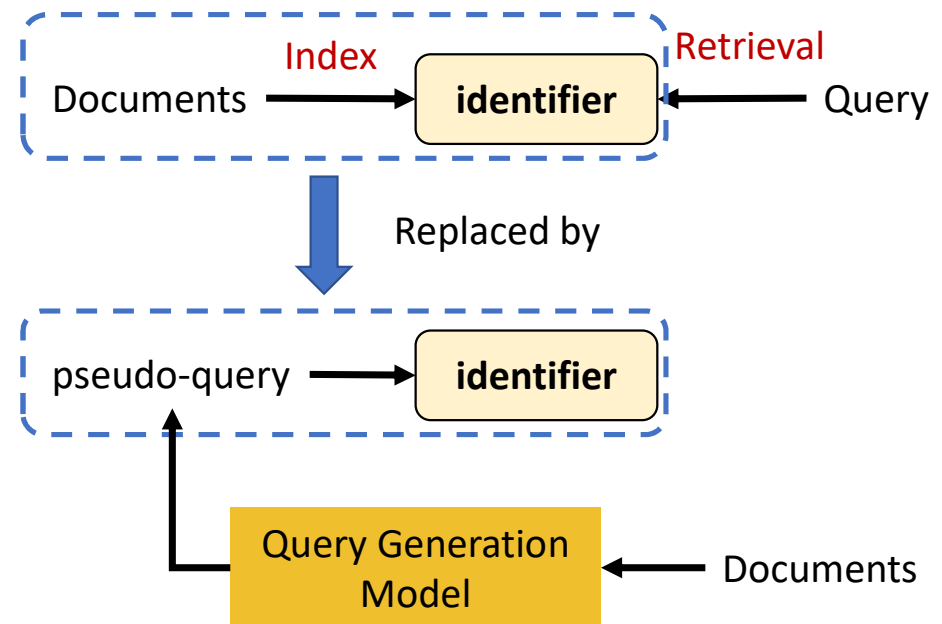
Limited supervised data limits the model to learn sufficient knowledge over each docid for retrieval and might lead to model over-fitting.

DSI-QG

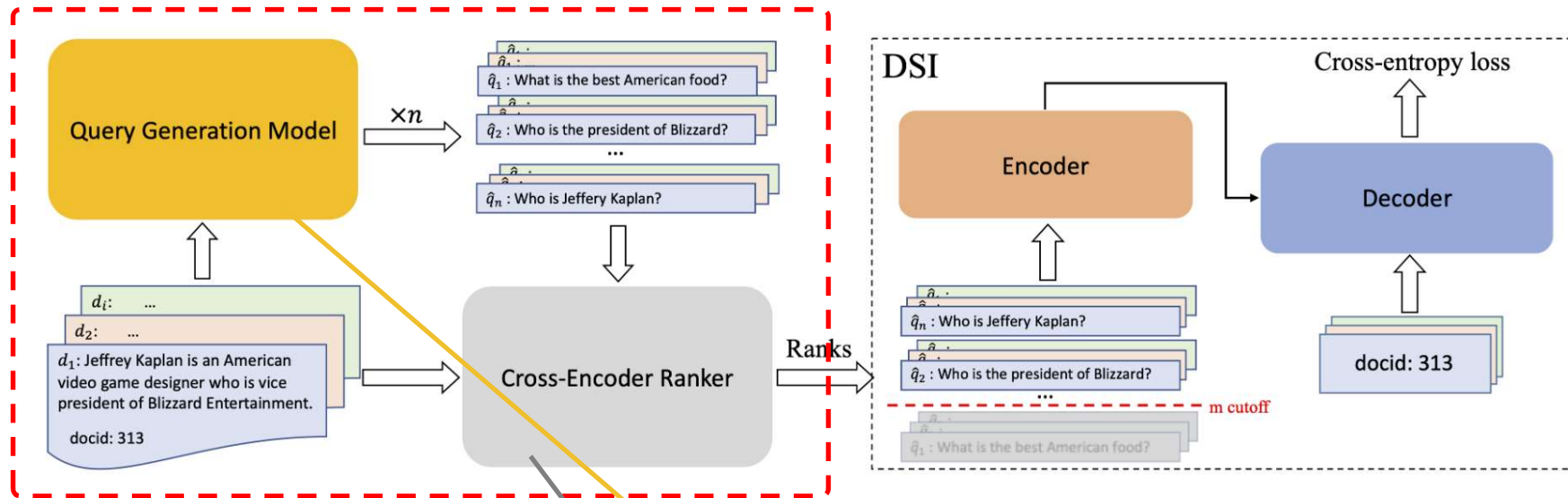
Motivation

Tackle the **data distribution mismatch** present between the **indexing** and the **retrieval** phases.

- Use a **query generation model** to generate a set of potentially-relevant queries to represent each candidate document for indexing.



DSI + Query Generation



- In addition to the **query generation model**, the framework also contains a **cross-encoder ranker**.
 - ❑ **Query Generation Model** is used to generate potential queries, which in turn are used to represent documents for indexing.
 - ❑ **Cross-Encoder Ranker** is used to select only promising queries to be included during indexing.

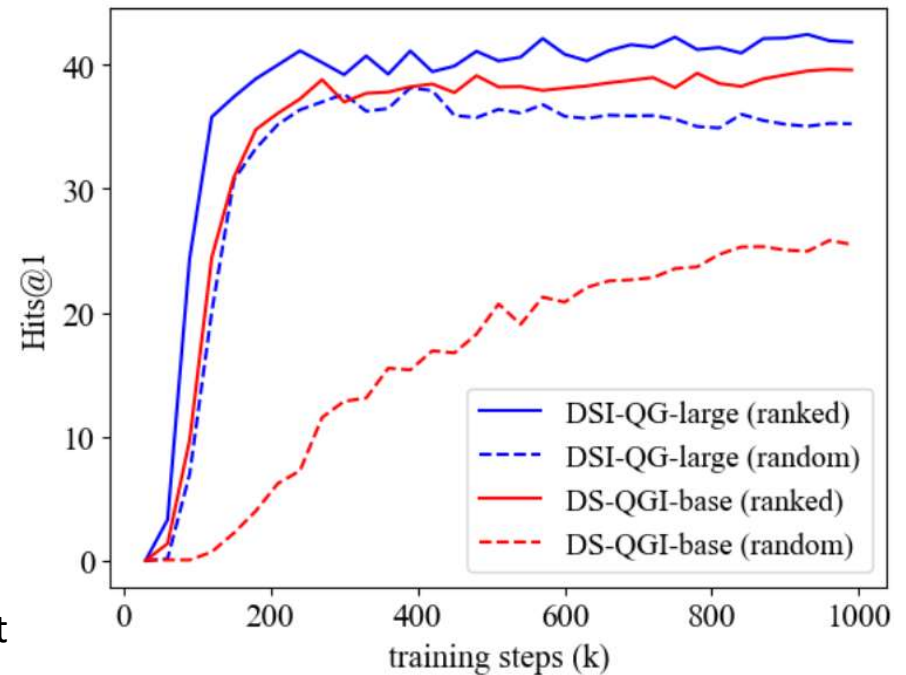
DSI + Query Generation



Impact of Cross-encoder Ranker

It still inevitably generates irrelevant queries due to the randomness of Query Generation Model and limited data.

- The experimental result with Cross-encoder Ranker yields higher Hit@1 convergence than the results without Cross-encoder Ranker.

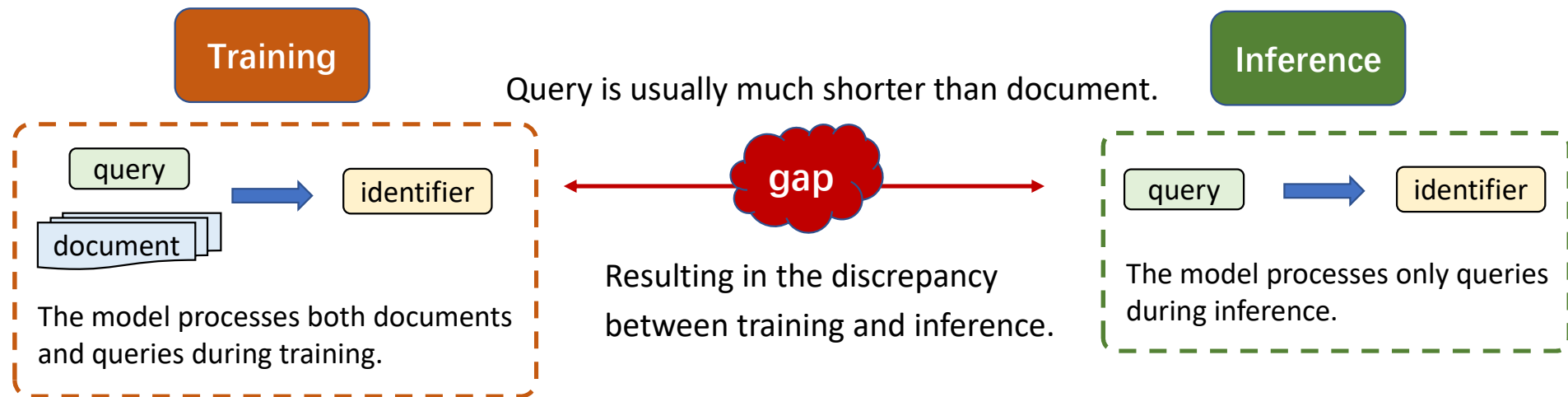




Training Strategy

- General Pre-train S-1
- Data Augmentation S-2
 - Query Generation Model
- Multi-stage Generation Architecture S-3
 - Two-stage Approach for Model-based Retrieval (TOME)
- Learning to Rank S-4

Two-stage generation



- **TOME: A Two-stage Approach for Model-based Retrieval**

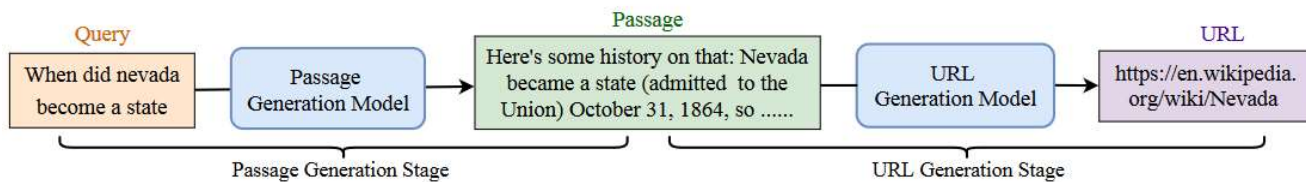
Motivation

Reduce the discrepancy between training and inference.

Two-stage generation

Motivation

Reduce the discrepancy between training and inference.



A two-stage generation approach with two different generation models:

- passage generation
- URL generation (Identifier generation)

Others

$\left\{ \begin{array}{l} \text{query} \rightarrow \text{identifier} \\ \text{passage} \rightarrow \text{identifier} \end{array} \right.$

TOME

query \rightarrow passage \rightarrow URL (identifier)

Two models handle two tasks separately, thus solving the problem of discrepancy.

Two-stage generation



Such a **two-stage generation approach** can more effectively **capture the semantic relatedness between queries and identifiers** by both reducing the training-inference discrepancy and enriching the generation context, which is specifically tailored for model-based retrieval.

Experimental results show:

❑ 95% of generated passages exist in the corpus. In cases where the model failed to generate relevant passages, about 85% of the generated passages still exist in the corpus.



- The model is **capable of memorizing the corpus** precisely and is able to generate a retrieval-like result.

❑ Although the passage generated by the passage generation model is not always exactly the same as the label passage, the URL generation model is still able to accurately map it to the correct URL.



- The URL generation model can **tolerate changes** introduced by the passage generation model.



Training Strategy

- General Pre-train S-1
- Data Augmentation S-2
 - Query Generation Model
- Multi-stage Generation Architecture S-3
 - Two-stage Approach for Model-based Retrieval (TOME)
- Learning to Rank S-4

Learning to Rank

Existing generative retrieval models

transform predicted identifiers into a passage rank list

There is a gap between the **learning objective of generative retrieval** and the desired **passage ranking target**



Motivation

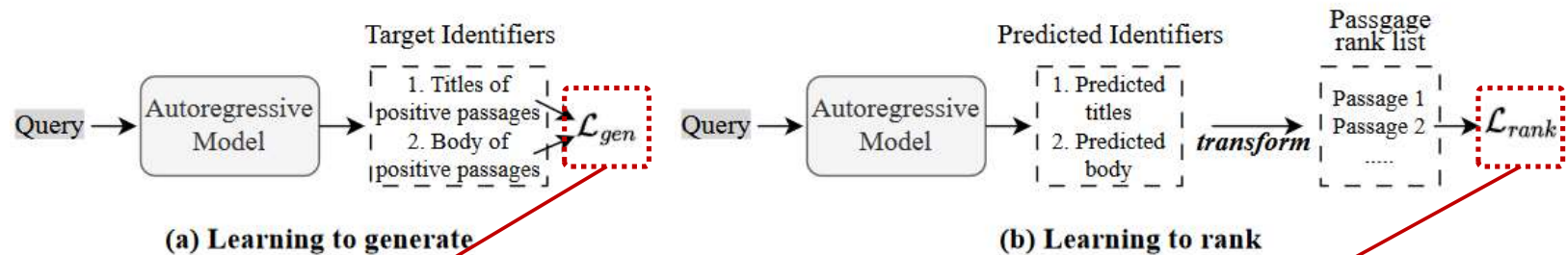
- Involve training an autoregressive model using a passage rank loss.
- Optimize the autoregressive model toward the optimal passage ranking.

Learning to Rank

- **LTRGR**: Combines generative retrieval with the classical learning-to-rank paradigm.

Two training stages:

- learning to generate
- learning to rank.



$$\mathcal{L}_{gen} = - \sum_{j=1}^l \log p_{\theta}(i_j | q; I_{<j}),$$

As previously mentioned, it is insufficient for generative retrieval to only learn how to generate identifiers.



$$\mathcal{L}_{rank} = \max(0, s(q, p_n) - s(q, p_p) + m),$$

Learning to Rank

$$s(q, p) = \sum_{i_p \in \mathcal{I}_p} s_{i_p}$$

$$\mathcal{L}_{rank} = \max(0, s(q, p_n) - s(q, p_p) + m),$$

- $S(q, p_n)$: The rank score of the passage p corresponding to the query q .
- S_{i_p} represents the language model score of the identifier i_p .
- I_p is the set of selected identifiers that appear in the passage p .

In practice, LTRGR used two rank losses based on the sampling strategy for positive and negative passages.

\mathcal{L}_{rank1} : the positive and negative passages are the ones with the **highest** rank scores.

\mathcal{L}_{rank2} : both the positive and negative passages are **randomly sampled** from the passage rank list.

Rank list

- Doc1 positive
- Doc2 positive
- Doc3 negative
- Doc4 positive
- Doc5 negative
-

Learning to Rank

The final loss

$$\mathcal{L} = \mathcal{L}_{rank1} + \mathcal{L}_{rank2} + \lambda \mathcal{L}_{gen}$$

Methods	Natural Questions		
	@5	@20	@100
w/o learning-to-rank	65.8	78.3	86.7
w/ rank loss 1	56.1	69.4	78.7
w/o generation loss	63.9	76.1	84.4
w/o rank loss	65.8	78.6	86.5
w/o rank loss 1	68.2	80.8	87.0
w/o rank loss 2	67.9	79.8	86.7
LTRGR	68.8	80.3	87.1

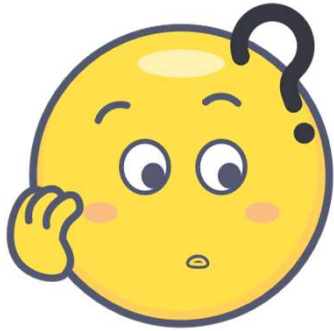
- ❑ Removing the generation loss is the only variant underperforming the original MINDER model. → **the necessity of the generation loss** in the learning-to-rank phase.
- ❑ Removing the rank causes performance drop. → the **learning-to-rank objective** is the primary source of performance improvement.
- ❑ Removing either \mathcal{L}_{rank1} or \mathcal{L}_{rank2} leads to a drop in the performance of LTRGR. → having two rank losses allows the model to **leverage a larger number of passages** and benefits the rank learning. And the two rank losses adopt different sample mining strategies, ensuring **the diversity of the passages** in the loss.



Outline

- Introduction to Retrieval Systems
- Three Main Challenges of Generative Retrieval
 - Identifier Design
 - Training Strategy
 - **Dynamic Corpora**
- Open Discussion

Dynamic Corpora



Incremental learning

how to develop Gen-IR systems that can adapt to **dynamic corpora** (i.e., how to **add** and **remove** **documents** from a model-indexed corpus)

Dynamic Corpora



Dynamic Corpora

Continuously indexing new documents while being able to answer queries related to both previously and newly indexed documents.

Dense retrieval



Generative retrieval

The contextualization and embedding capability of bi-encoders were considered sufficient and **adaptable to updates without model parameter updates** in many cases, so model parameter does **not need to be updated frequently**.

It needs to be **re-trained from scratch** every time the underlying corpus is updated because all documents are **fully parameterized** and thus susceptible to overfitting the train data, **being less adaptable to updates**.

Higher computational cost

Dynamic Corpora

- **StreamingIR:**

A realistic retrieval benchmark on temporal knowledge

		Total (2007 – 2020)								Past (2007 – 2019)								Recent (2020)							
		Hits@N				Answer Recall@N				Hits@N				Answer Recall@N				Hits@N				Answer Recall@N			
		5	10	50	100	5	10	50	100	5	10	50	100	5	10	50	100	5	10	50	100	5	10	50	100
only fine-tuning without pretraining																									
bi	ft	24.8	30.5	44.5	49.1	44.6	53.3	69.7	73.9	17.9	22.1	34.5	38.6	35.7	44.6	63.0	68.3	29.4	36.1	51.2	56.1	50.5	59.1	74.1	77.7
gr	ft	13.8	16.6	22.8	25.1	37.4	43.8	55.8	60.0	32.4	39.1	54.1	60.0	55.6	63.1	77.1	79.9	1.4	1.6	2.0	2.1	25.3	30.9	41.7	46.7

GR fails with only retrieval fine-tuning

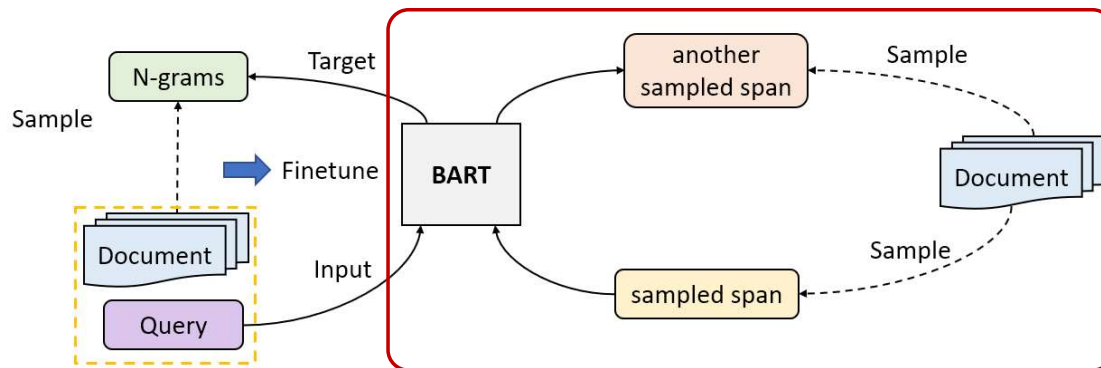
The inherent limitation of generative retrieval models

- ❑ The performance of generative retrieval models **deteriorates heavily when evaluated from recent subsets**, which requires to correctly retrieve relevant information from the most recent knowledge (2020).
- ❑ Bi-encoder shows consistent performance to both past and recent subsets of the evaluated queries.

Dynamic Corpora

		Total (2007 – 2020)								Past (2007 – 2019)								Recent (2020)							
		Hits@N				Answer Recall@N				Hits@N				Answer Recall@N				Hits@N				Answer Recall@N			
		5	10	50	100	5	10	50	100	5	10	50	100	5	10	50	100	5	10	50	100	5	10	50	100
only fine-tuning without pretraining																									
bi	ft	24.8	30.5	44.5	49.1	44.6	53.3	69.7	73.9	17.9	22.1	34.5	38.6	35.7	44.6	63.0	68.3	29.4	36.1	51.2	56.1	50.5	59.1	74.1	77.7
gr	ft	13.8	16.6	22.8	25.1	37.4	43.8	55.8	60.0	32.4	39.1	54.1	60.0	55.6	63.1	77.1	79.9	1.4	1.6	2.0	2.1	25.3	30.9	41.7	46.7
unsupervised (only pretraining)																									
bi	2007 – 2019	20.5	25.3	38.2	41.5	37.4	44.8	61.6	66.0	6.7	8.5	16.0	17.6	20.0	26.4	44.8	50.3	29.7	36.6	53.1	57.5	49.0	57.2	72.9	76.5
	+ 2020	19.5	24.3	37.5	41.8	39.2	47.8	65.6	70.1	12.3	15.8	26.5	30.4	30.1	38.6	57.8	63.3	24.7	30.5	45.3	50.0	46.2	54.8	71.6	75.3
gr	2007 – 2019	34.5	38.9	49.2	52.8	47.2	54.5	68.8	73.3	32.7	37.5	48.9	52.4	45.3	52.5	68.8	73.3	35.6	39.9	49.4	53.1	48.5	55.8	68.9	73.3
	+ 2020	34.3	38.9	49.0	52.8	47.2	54.5	68.6	73.1	32.7	37.5	48.7	52.2	45.2	52.8	68.6	73.0	35.5	39.8	49.3	53.1	48.5	55.6	68.6	73.2

SEAL



Unsupervised learning can mitigate this problem of Generative Retrieval, but its effectiveness is limited, and further consideration of other measures is still necessary.

Dynamic Corpora



Parameter-efficient method can help generative retrievers to updated knowledge?

The full parameter approach forget heavily on existing knowledge.

LoRA exhibits no sign of forgetting and excels in acquiring new knowledge

	Total (2007 – 2020)								Past (2007 – 2019)								Recent (2020)							
	Hits@N				Answer Recall@N				Hits@N				Answer Recall@N				Hits@N				Answer Recall@N			
	5	10	50	100	5	10	50	100	5	10	50	100	5	10	50	100	5	10	50	100	5	10	50	100
supervised (pretrain + fine-tuned)																								
bi 2007 – 2019 + ft	40.1	47.6	63.3	67.4	59.0	67.5	81.0	83.7	29.7	35.8	51.9	57.2	49.5	59.0	76.5	80.7	47.1	55.4	71.0	74.2	65.4	73.1	84.0	85.6
gr 2007 – 2019	33.2	39.8	54.1	59.5	55.7	63.5	77.3	81.2	32.8	38.9	54.5	60.6	56.8	64.5	79.0	82.8	33.5	40.4	53.8	58.9	54.9	62.9	76.2	80.2
+ full pretrain + ft	31.7	37.1	51.1	56.1	51.7	59.5	72.6	76.1	29.9	35.8	49.6	54.8	50.9	59.0	72.2	76.1	32.9	38.1	52.1	57.0	52.3	59.9	72.8	76.1
+ LoRA (att.) + ft	33.3	39.7	54.0	59.5	54.7	62.8	75.7	79.3	32.6	39.2	54.2	60.4	55.4	64.1	76.7	80.6	33.7	40.1	53.9	58.9	54.2	61.9	75.0	78.4
+ LoRA (exp.) + ft	38.2	44.1	58.5	64.4	59.7	67.4	80.3	83.9	37.3	43.2	58.3	64.4	59.2	67.7	81.3	85.2	38.9	44.8	58.7	64.5	60.0	67.2	79.6	83.0
+ ft w/ 2020 Q	41.6	48.5	63.4	68.3	62.9	70.7	81.9	85.2	38.3	45.3	60.2	65.0	60.0	68.1	80.3	84.1	43.8	50.7	65.6	70.6	64.9	72.4	83.0	85.9

- Lora allows the model to retain existing knowledge while acquiring new knowledge

Dynamic Corpora

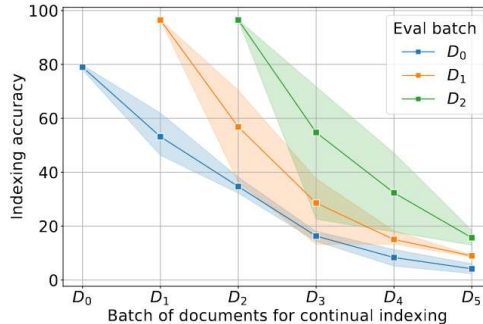
DSI++ conducted a more in-depth analysis of this issue.



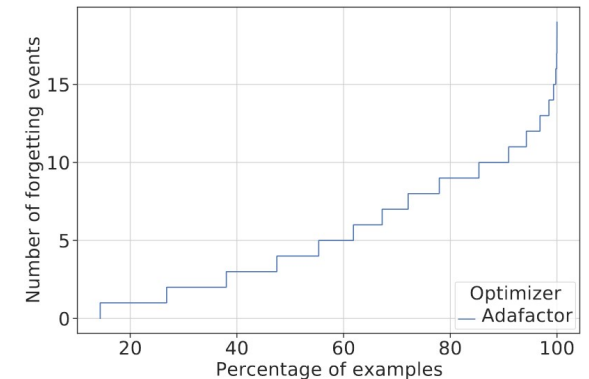
A naive solution for Dynamic Corpora is to **continuously fine-tune the model** with an indexing objective over new documents. But continual indexing of new documents leads to **catastrophic forgetting** of the previously memorized documents.

Here are two key challenges to overcome for Dynamic Corpora.

Explicit forgetting from continual indexing of new documents



Implicit forgetting during memorization



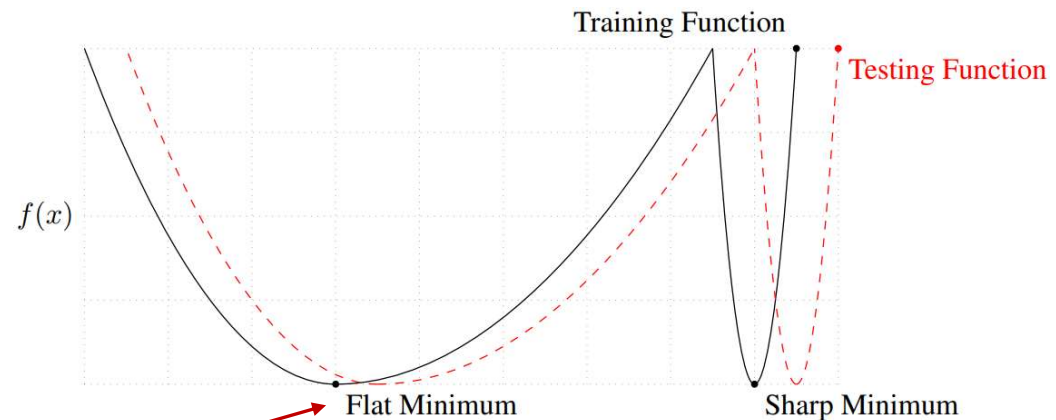
- Continual indexing of new documents leads to catastrophic forgetting of the previously memorized documents.

- A significant number of documents experience forgetting events (individual documents go from being classified correctly to incorrectly) after they have been memorized.

Sharpness-Aware Minimization

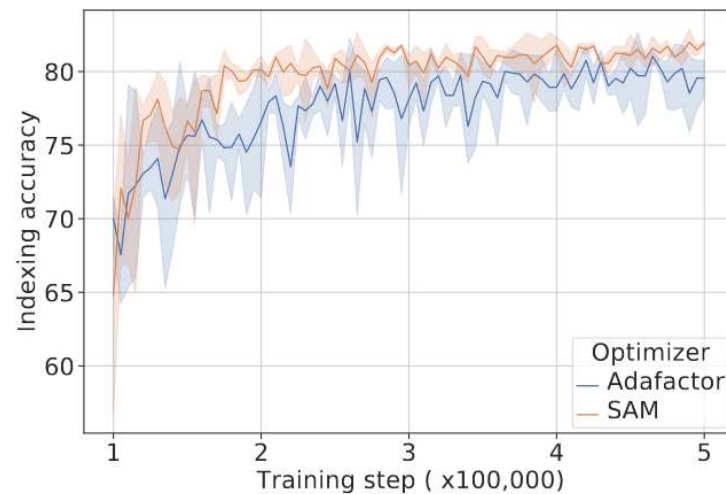
Implicit forgetting

- **Sharpness-Aware Minimization (SAM)** is an optimization algorithm for training neural networks. Its key idea is to **reduce the sharpness of the loss function** in parameter space, aiming to **improve model generalization**.

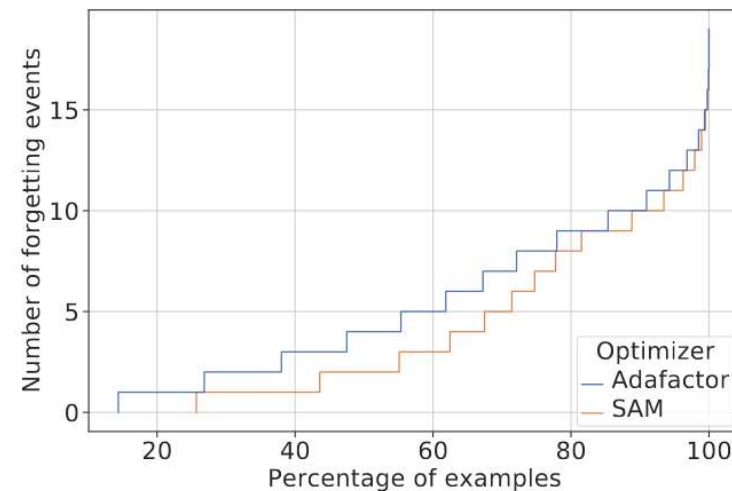


Introducing a regularization term based on the gradient norm of the loss in parameter space, encouraging the model to move towards **flatter regions** rather than sharp local minima during training.

Sharpness-Aware Minimization



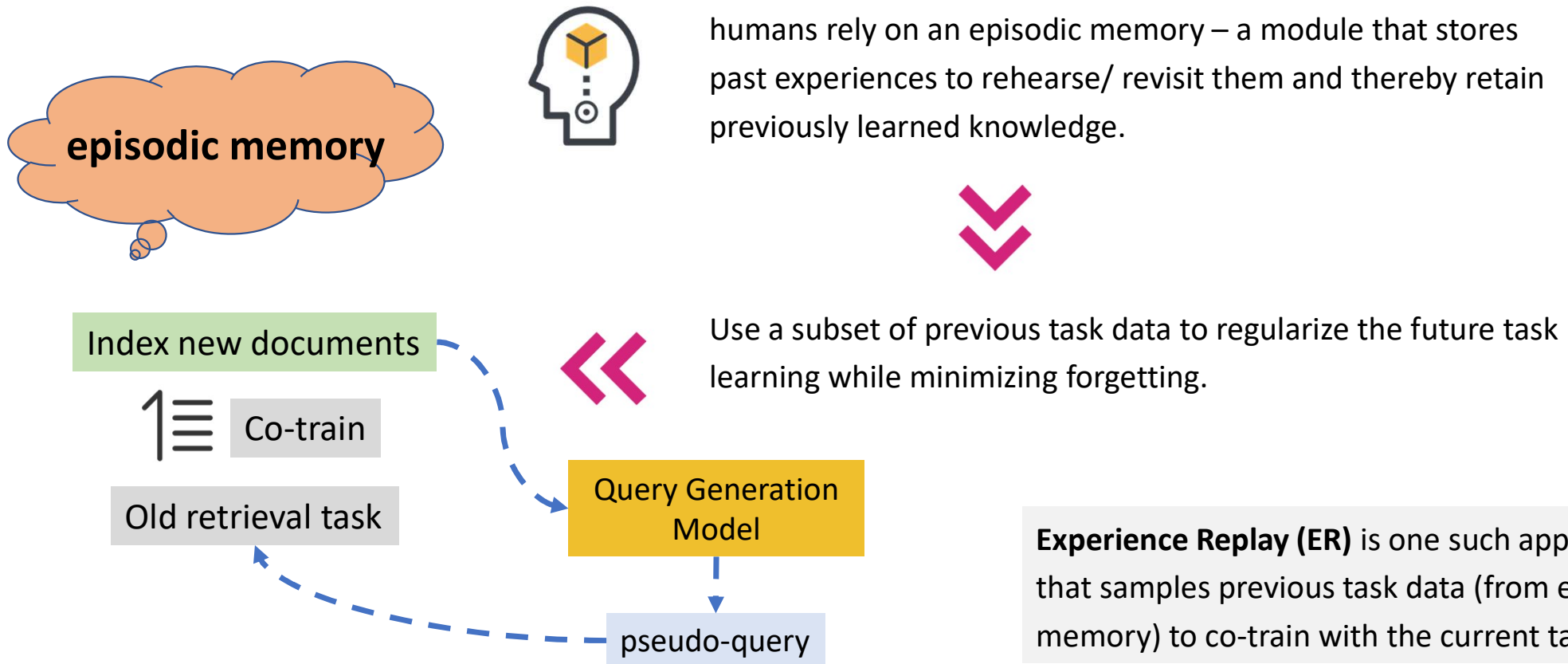
- SAM leads to relatively **stable memorization** of the documents.



- SAM **increases** the percentage of examples experiencing **zero forgetting events** by absolute 12% over Adafactor.

Generative memory

Explicit forgetting



Generative memory

Added corpus	Method	Eval corpus = D_0 (Catastrophic forgetting)			Eval corpus = D_1 (Forward transfer)		
		Index acc.	Hits@1	Hits@10	Index acc.	Hits@1	Hits@10
<i>Natural Questions (NQ)</i> - $ D_0 = 50K, D_1 = 10K$							
D_0	-	81.8 _{1.2}	35.9 _{2.2}	66.9 _{0.9}	-	-	-
D_1	cl(D_1)	52.4 _{3.5}	19.2 _{3.9}	43.6 _{5.7}	96.5 _{0.0}	31.7 _{6.4}	55.6 _{4.9}
	cl($U_1 = D_0 \cup D_1$)	78.2 _{0.5}	28.9 _{8.9}	59.0 _{7.9}	91.8 _{0.4}	34.0 _{2.4}	60.2 _{1.9}
	cl(U_1)+epsmem(D_0)	77.8 _{0.5}	22.9 _{1.5}	51.4 _{0.5}	93.1 _{0.0}	13.1 _{2.1}	39.6 _{3.1}
	cl(U_1)+genmem(D_0)	77.8 _{0.3}	26.0 _{6.9}	54.9 _{8.3}	93.0 _{0.5}	8.6 _{4.8}	31.6 _{11.8}
	cl(U_1)+epsmem(D_1)	53.2 _{3.1}	7.7 _{2.1}	26.0 _{2.0}	96.5 _{0.0}	48.3 _{2.3}	70.7 _{1.9}
	cl(U_1)+genmem(D_1)	50.1 _{0.8}	7.0 _{1.2}	23.1 _{2.2}	96.5 _{0.0}	57.7 _{1.5}	76.7 _{0.9}
	cl(U_1)+genmem(U_1)	78.2 _{0.3}	18.4 _{2.8}	47.5 _{3.9}	92.1 _{0.3}	48.5 _{6.1}	73.8 _{2.9}
train from scratch		78.7 _{0.6}	35.9 _{1.4}	66.4 _{0.0}	79.2 _{0.3}	32.9 _{1.8}	63.9 _{1.2}
<i>MS MARCO (full)</i> - $ D_0 = 8M, D_1 = 842K$							
D_0	-	99.4	16.3	46.8	-	-	-
D_1	cl(D_1)	0.0	0.1	0.6	97.9	18.2	40.5
	cl(U_1)+genmem(U_1)	20.4	7.3	31.3	86.6	31.6	65.8

- Experience Replay (ER) with the generative memory improves the overall performance for the retrieval task, **reducing forgetting** of previously indexed documents and enabling forward transfer to newly indexed documents.
- This approach outperforms re-training the model from scratch in terms of overall performance and is **computationally efficient**.

- cl(U_1): continue fine-tuning with indexing task on U_1
- cl(U_1)+epsmem(D): continual indexing of U_1 along with ER of queries for D
- cl(U_1)+genmem(D): continual indexing of U_1 along with ER of pseudo-queries for D

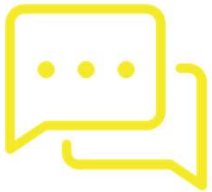
Mehta S V, Gupta J, Tay Y, et al. DSI++: Updating Transformer Memory with New Documents[J]. arXiv preprint arXiv:2212.09744, 2022.



Outline

- Introduction to Retrieval Systems
- Three Main Challenges of Generative Retrieval
 - Identifier Design
 - Training Strategy
 - Dynamic Corpora
- **Open Discussion**

Open Discussion



- (i) It has yet to be demonstrated that retrieval performance is improved on **Large scale corpus** (such as the full MS-MARCO dataset).
- (ii) For generative retrieval models, there remain open questions about the practical applicability of such models to **dynamic corpora**. It is valuable to explore continuously updated learning objectives over new or removed documents.
- (iii) Traditional index-retrieve-then-rank paradigm implies a **Learning To Rank objective** at the end of the pipeline. How to further integrate generative retrieval with the classical learning-to-rank paradigm is also an interesting direction.
- (iv) **Bridging the gap of IR and NLP**. Thanks to the powerful contextual modeling capabilities of LLMs, GR models are able to find documents that are related to queries **in a more associative & nuanced way**.

Large scale corpus

Model	MSMarco100k			MSMarco1M			MSMarcoFULL		
	At.	Nv.	Sm.	At.	Nv.	Sm.	At.	Nv.	Sm.
<i>Baselines</i>									
BM25	-	65.3	-	-	41.3	-	-	18.4	-
BM25 (w/ doc2query-T5)	-	80.4	-	-	56.6	-	-	27.2	-
GTR-Base	-	83.2	-	-	60.7	-	-	34.8	-
<i>Ours</i>									
(1a) Labeled Queries (No Indexing)	0.0	1.1	0.0	0.0	0.5	0.0	0.0	0.0	0.0
(2a) FirstP/DaQ + Labeled Queries (DSI)	0.0	23.9	19.2	2.1	12.4	7.4	0.0	7.5	3.1
(3b) FirstP/DaQ + D2Q + Labeled Queries	79.2	77.7	76.8	53.3	48.2	47.1	14.2	13.2	6.4
(4a) 3b + PAWA (w/ 2D Semantic IDs)	-	-	77.1	-	-	50.2	-	-	9.0
(5) 4a + Consistency Loss (NCI)	-	-	77.1	-	-	50.2	-	-	9.1
(6b) D2Q only	80.3	78.7	78.5	55.8	55.4	54.0	24.2	13.3	11.8
(4a') 6b + PAWA (w/ 2D Semantic IDs)	-	-	78.2	-	-	54.1	-	-	17.3
(4b') 6b + Constrained Decoding	-	-	78.6	-	-	54.0	-	-	12.0
(5') 6b + PAWA (w/ 2D Semantic IDs) + Constrained Decoding	-	-	78.3	-	-	54.2	-	-	17.4



- As corpus size grows, the performance of all methods rapidly drops off.
- Some methods can be used to alleviate this problem, such as using **semantic id, PAWA decoding, increasing model scale**, etc.

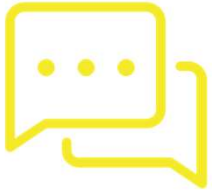
Large scale corpus

T5 Scale	Training	Params	Inference FLOPs	MRR@10
Base	D2Q Only + Atomic ID	7.0B	0.9×10^{12}	24.2
Base	D2Q Only + Naive ID	220M	1.4×10^{12}	13.3
Base	D2Q Only + PAWA (2D Sem.)	761M	6.8×10^{12}	17.3
Large	D2Q Only + Naive ID	783M	3.5×10^{12}	21.4
Large	D2Q Only + PAWA (2D Sem.)	2.1B	1.1×10^{13}	19.8
XL	D2Q Only + Naive ID	2.8B	9.3×10^{12}	26.7
XXL	D2Q Only + Naive ID	11B	4.3×10^{13}	24.3



Further scaling the model size does not always improve performance.

Open Discussion



- (i) It has yet to be demonstrated that retrieval performance is improved on **Large scale corpus** (such as the full MS-MARCO dataset).
- (ii) For generative retrieval models, there remain open questions about the practical applicability of such models to **dynamic corpora**. It is valuable to explore continuously updated learning objectives over new or removed documents.
- (iii) Traditional index-retrieve-then-rank paradigm implies a **Learning To Rank objective** at the end of the pipeline. How to further integrate generative retrieval with the classical learning-to-rank paradigm is also an interesting direction.
- (iv) **Bridging the gap of IR and NLP**. Thanks to the powerful contextual modeling capabilities of LLMs, GR models are able to find documents that are related to queries **in a more associative & nuanced way**.

Thank you for your attention!



Pengjie Ren
renpengjie@sdu.edu.cn
<https://pengjieren.github.io/>