

# Generative Information Retrieval



SIGIR-AP 2023 tutorial

---

**Yubao Tang<sup>a,b</sup>, Ruqing Zhang<sup>a,b</sup>, Jiafeng Guo<sup>a,b</sup> and Maarten de Rijke<sup>c</sup>**

<https://sigir-ap2023-generative-ir.github.io/>

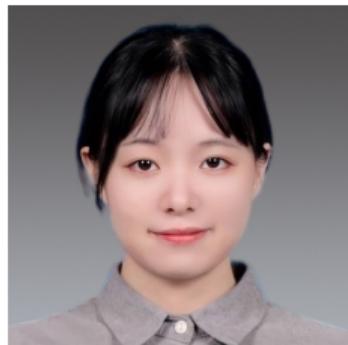
November 26, 2023

<sup>a</sup> Institute of Computing Technology, Chinese Academy of Sciences

<sup>b</sup> University of Chinese Academy of Sciences

<sup>c</sup> University of Amsterdam

## About presenters



Yubao Tang

Phd student  
@ICT, CAS



Ruqing Zhang

Faculty  
@ICT, CAS



Jiafeng Guo

Faculty  
@ICT, CAS



Maarten de Rijke

Faculty  
@UvA

# Information retrieval

Information retrieval (IR) is the activity of obtaining information system resources that are relevant to an information need from a collection of those resources.

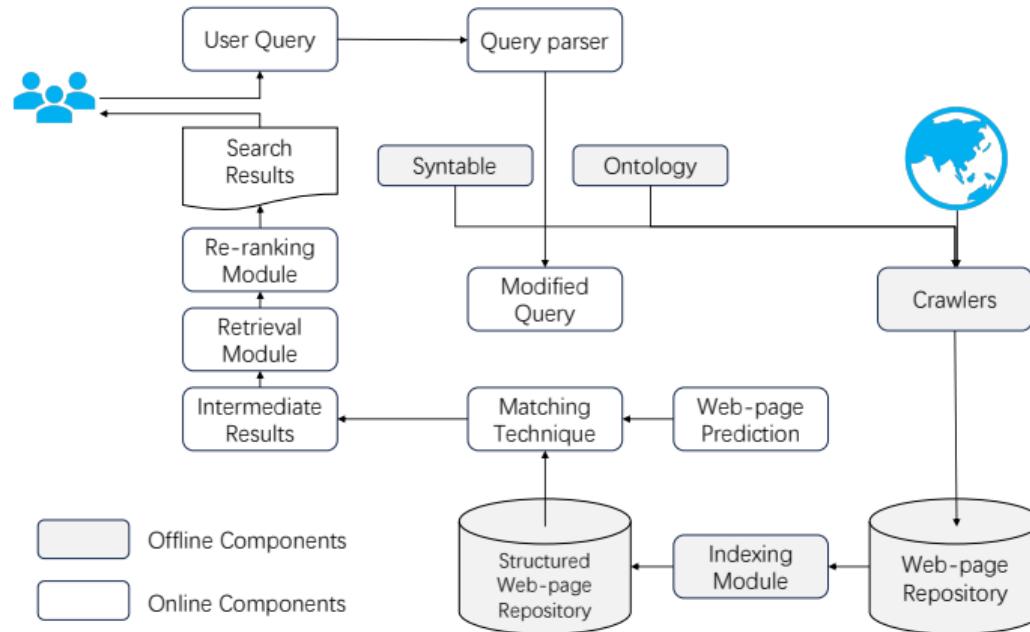


**Given:** User query (keywords, question, image, ...)

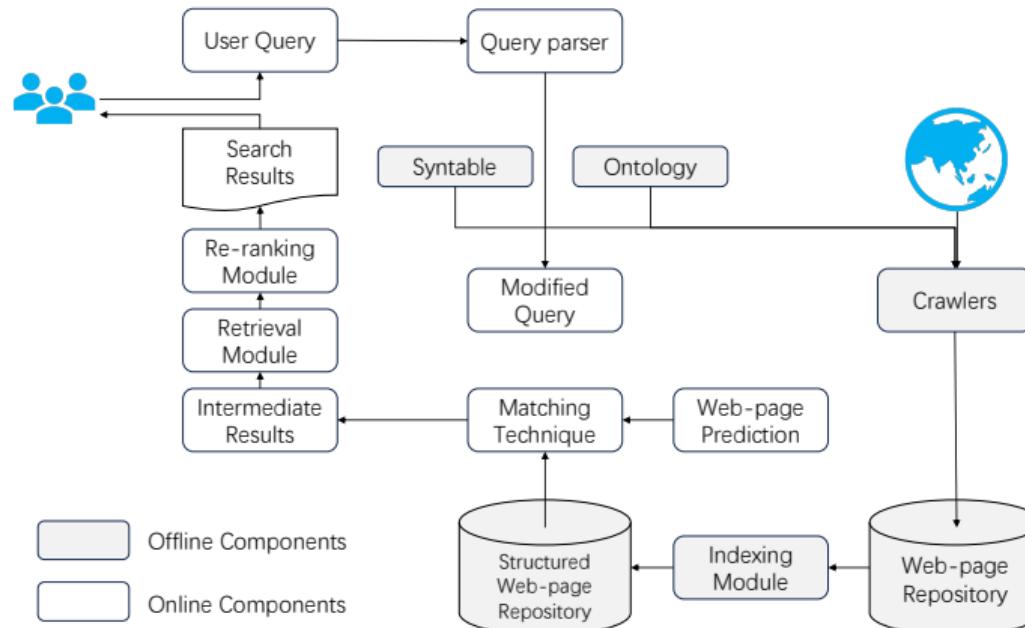
**Rank:** Information objects (passages, documents, images, products, ...)

**Ordered by:** Relevance scores

# Complex architecture design behind search engines

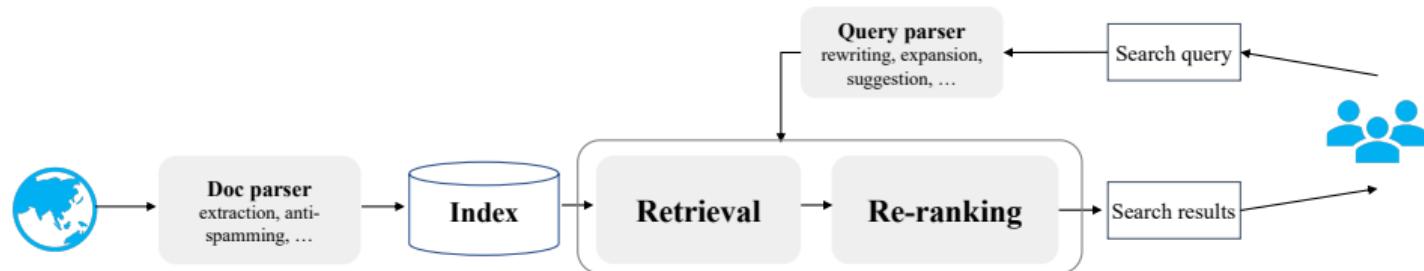


# Complex architecture design behind search engines



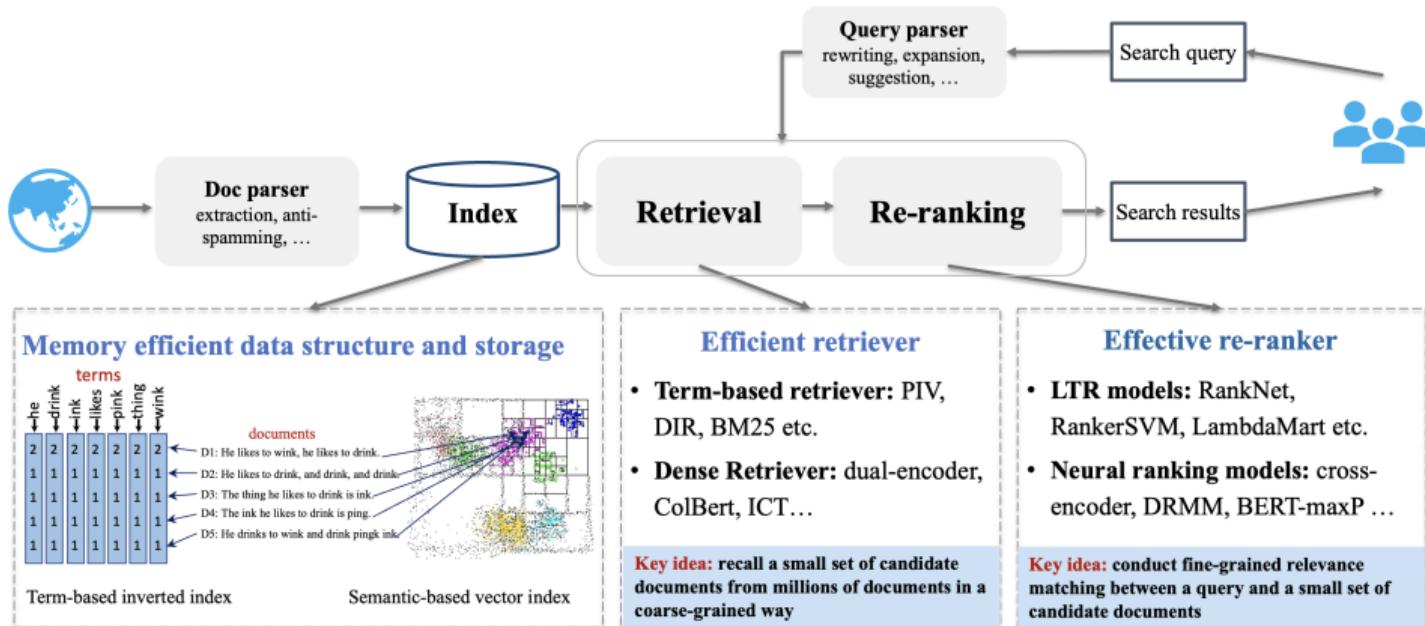
- **Advantages:**
  - Pipelined paradigm has withstood the test of time
  - Advanced machine learning and deep learning approaches applied to many components of modern systems

# Core pipelined paradigm: Index-Retrieval-Ranking



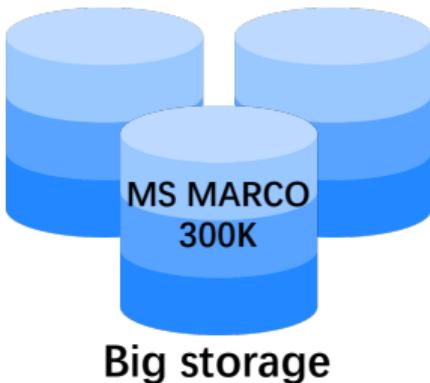
- Index: Build an index for each document in the entire corpus
- Retriever: Find an initial set of candidate documents for a query
- Re-ranker: Determine the relevance degree of each candidate

# Index-Retrieval-Ranking: Disadvantages



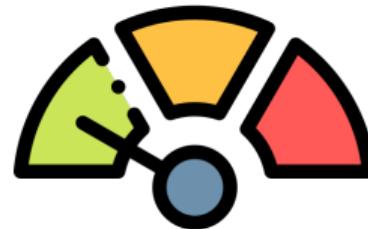
- **Effectiveness:** Heterogeneous ranking components are usually difficult to be optimized in an end-to-end way towards the global objective

## Index-Retrieval-Ranking: Disadvantages



**Big storage**

GTR (Dense retrieval)  
Memory size **1430MB**



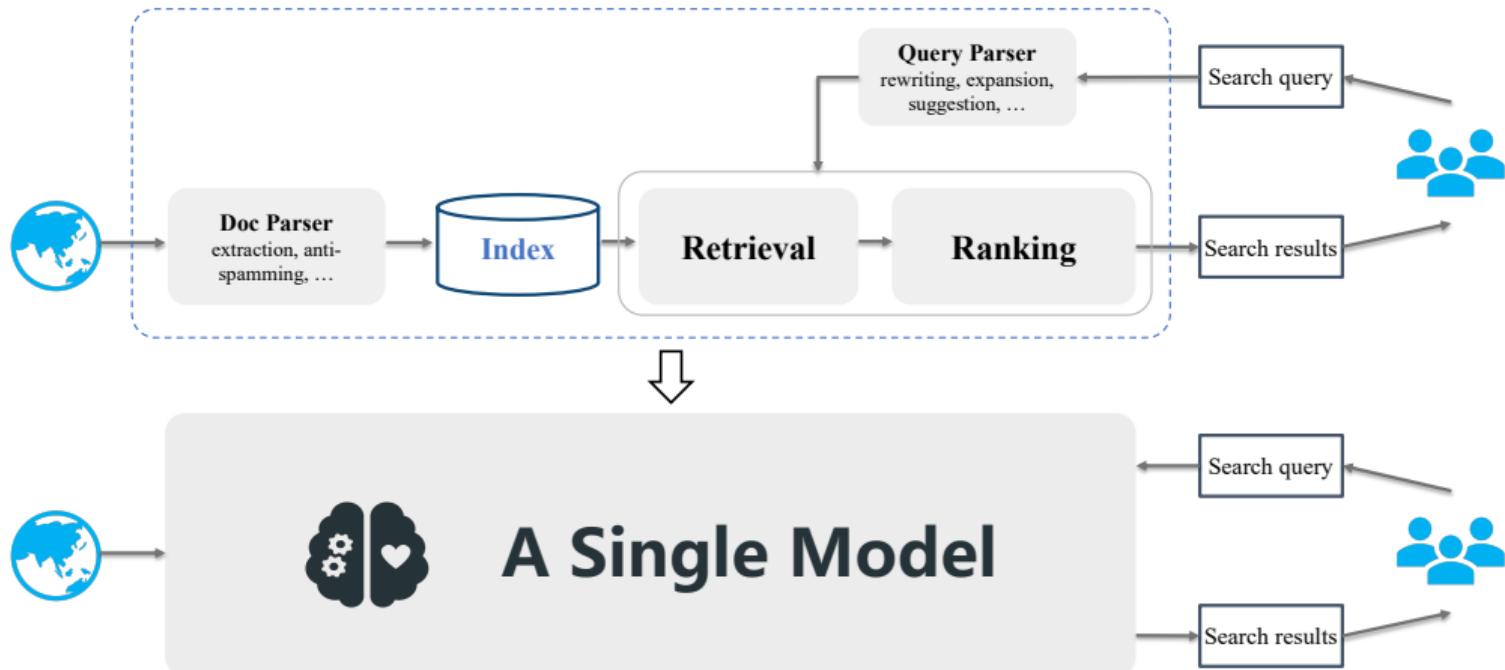
**Slow inference speed**

GTR (Dense retrieval)  
Online latency **1.97s**

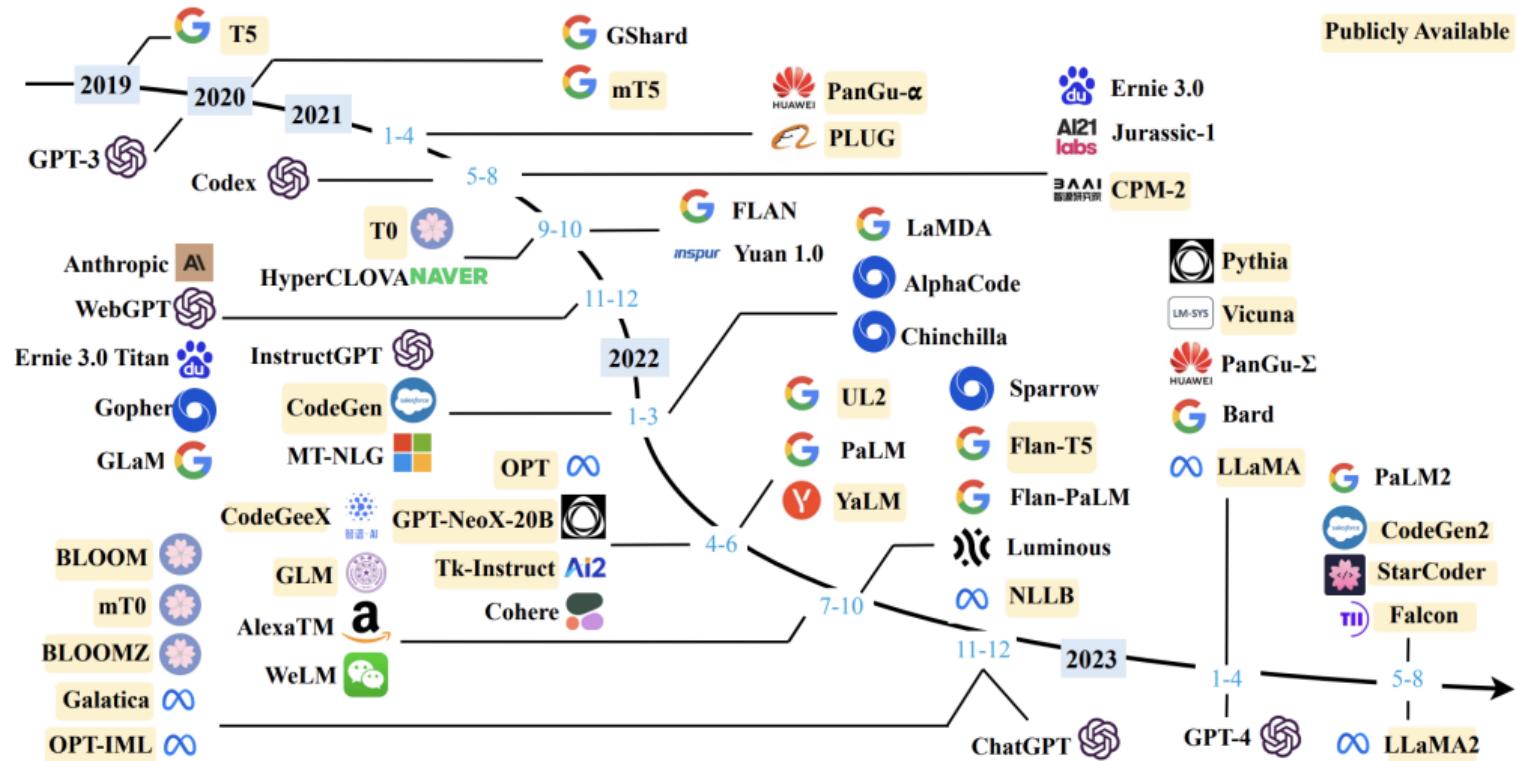
- **Efficiency:** A large document index is needed to search over the corpus, leading to significant memory consumption and computational overhead

**What if we replaced the pipelined architecture with a single consolidated model that efficiently and effectively encodes all of the information contained in the corpus?**

# Opinion paper: A single model for IR



# Generative language models



## Two families of generative retrieval

- **Closed-book**: The language model is the **only source** of knowledge leveraged during generation, e.g.,
  - Capturing document ids in the language models
  - Language models as retrieval agents via prompting
- **Open-book**: The language model can draw on **external memory** prior to, during and after generation, e.g.,
  - Retrieve-augmented generation of answers
  - Tool-augmented generation of answers

## Two families of generative retrieval

- **Closed-book**: The language model is the **only source** of knowledge leveraged during generation, e.g.,
  - Capturing document ids in the language models
  - Language models as retrieval agents via prompting
- **Open-book**: The language model can draw on **external memory** prior to, during and after generation, e.g.,
  - Retrieve-augmented generation of answers
  - Tool-augmented generation of answers

## Closed-book generative retrieval

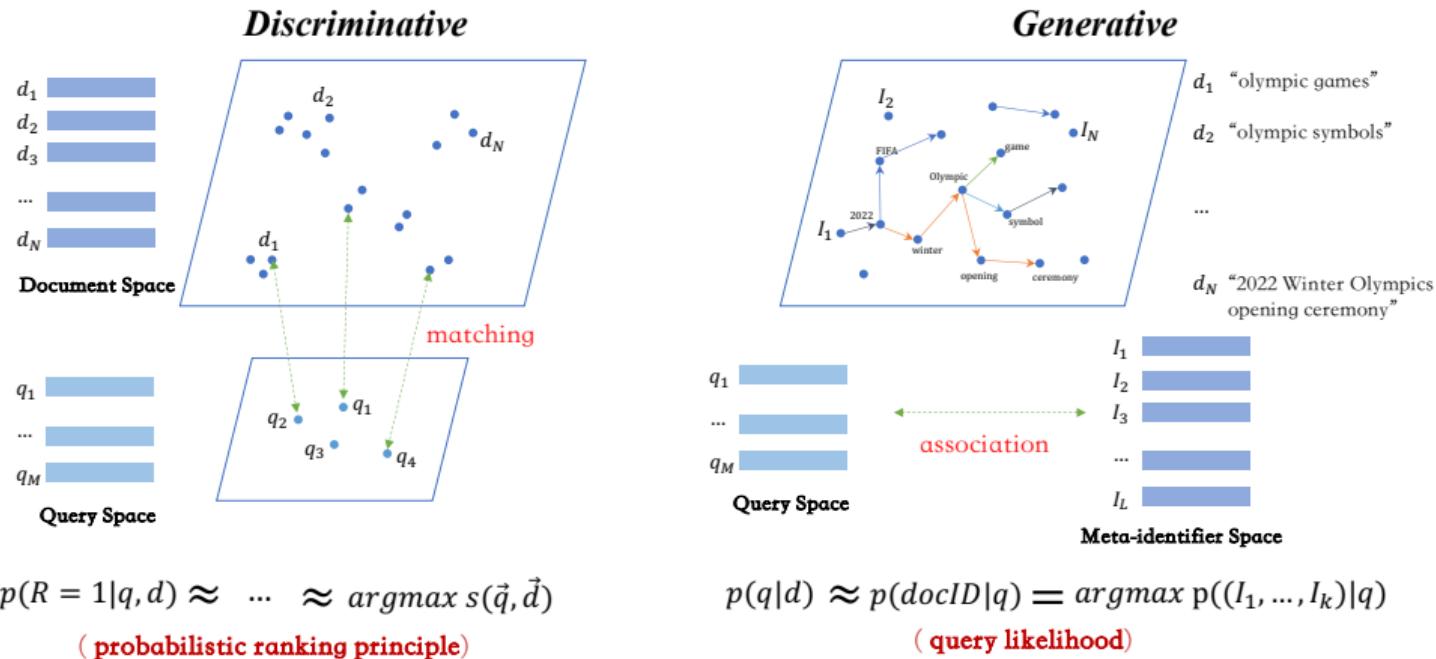
The IR task can be formulated as a **sequence-to-sequence (Seq2Seq)** generation problem

## Closed-book generative retrieval

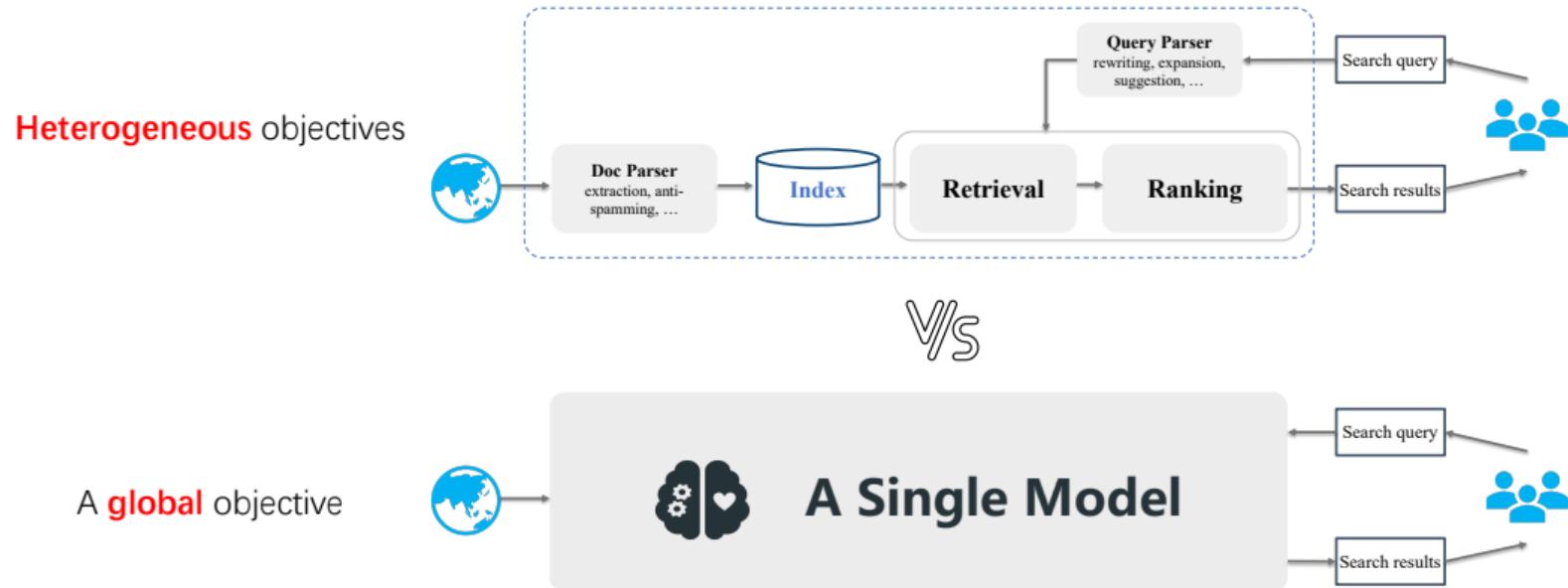
The IR task can be formulated as a **sequence-to-sequence (Seq2Seq)** generation problem

- **Input:** A sequence of query words
- **Output:** A sequence of document identifiers

# Neural IR models: Discriminative vs. Generative

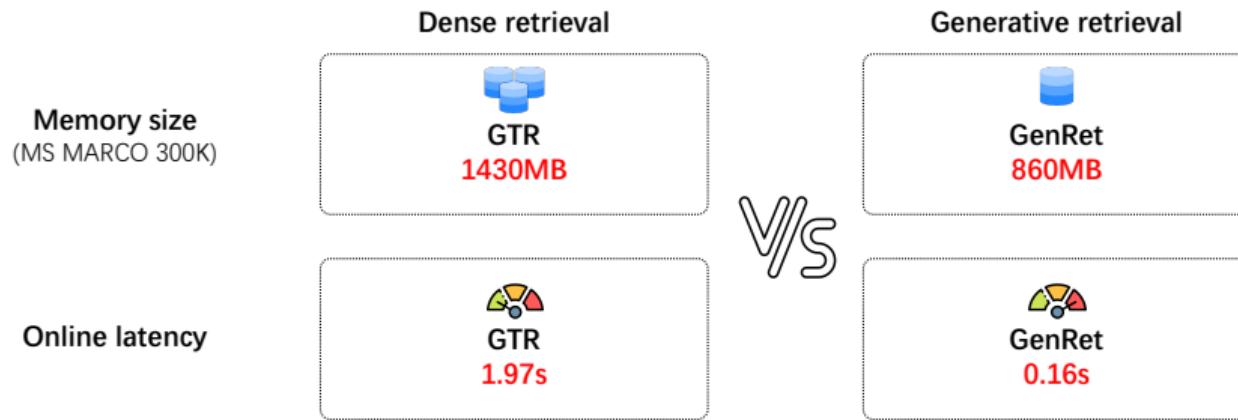


# Why generative retrieval?



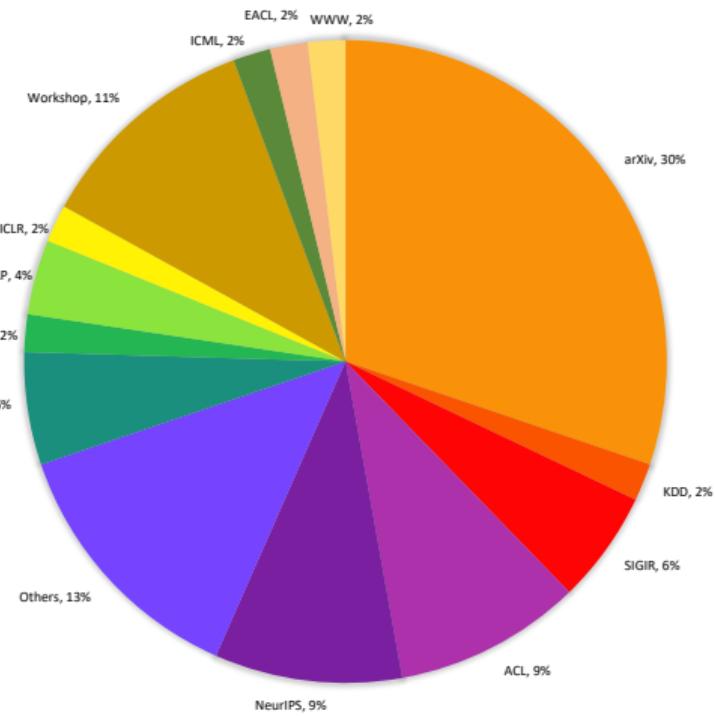
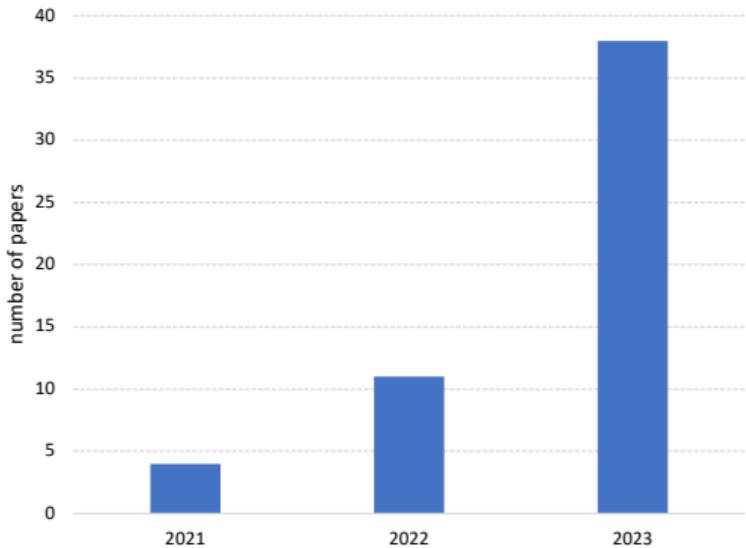
- **Effectiveness:** Knowledge of all documents in corpus is encoded into model parameters, which can be optimized directly in an end-to-end manner

# Why generative retrieval?



- **Efficiency:** Main memory computation of GR is the storage of document identifiers and model parameters
- Heavy retrieval process is replaced with a light generative process over the vocabulary of identifiers

# Statistics of related publications



The data statistics cover up to November 20, 2023.

# The First Workshop on Generative Information Retrieval @SIGIR2023

Title	Description	Panelists	Moderator	Start	End
1 Opening			Gabriel Bénédict	Thursday, 9:00 AM	Thursday, 9:30 AM
2 Panel Discussions	<b>Model Training</b> Architectures, training, RL, etc.	Jiafeng Guo, Vinh Q. Tran Minjoon Seo, Rajhans Samdani	Don Metzler	Thursday, 9:30 AM	Thursday, 10:30 AM
3 Coffee Break				Thursday, 10:30 AM	Thursday, 11:00 AM
4 Poster Session	Shared with REML and ReNeuIR			Thursday, 11:00 AM	Thursday, 12:30 PM
5 Lunch				Thursday, 12:30 PM	Thursday, 1:30 PM
6 Panel Discussions	<b>Broader Issues</b> Evaluation, HCI, ecosystem concerns, societal issues, etc.	Chirag Shah, Emily Bender Yiqun Liu, Guido Zuccon	Gabriel Bénédict, RZ Ruqing Zhang	Thursday, 1:30 PM	Thursday, 2:30 PM
7 Coffee Break				Thursday, 2:30 PM	Thursday, 2:45 PM
8 Panel Discussions	<b>Model Behavior</b> Inputs, outputs, hallucination, answer generation, etc.	Chua Tat-Seng, Omar Khattab Nazneen Fatema Rajani Fabio Petroni	Andrew Yates, RZ Ruqing Zhang	Thursday, 2:45 PM	Thursday, 3:45 PM
9 Coffee Break				Thursday, 3:45 PM	Thursday, 4:00 PM
10 Roundtable Discussion			Gabriel Bénédict, Andrew Yates	Thursday, 4:00 PM	Thursday, 4:45 PM
11 Closing			Gabriel Bénédict	Thursday, 4:45 PM	Thursday, 5:00 PM

<https://coda.io/@sigir/gen-ir>

## Goals of the tutorial

- We will cover key developments on generative information retrieval (mostly 2021–2023)
  - **Problem definitions**
  - **Docid design**
  - **Training approaches**
  - **Inference strategies**
  - **Applications**

## Goals of the tutorial

- We will cover key developments on generative information retrieval (mostly 2021–2023)
  - **Problem definitions**
  - **Docid design**
  - **Training approaches**
  - **Inference strategies**
  - **Applications**
- We are still far from understanding how to best develop generative IR architecture compared to traditional pipelined IR architecture:
  - Taxonomies of existing research and key insights
  - Our perspectives on the **current challenges & future directions**

# Schedule

Time	Section	Presenter
13:00-13:10	Section 1: Introduction	Maarten de Rijke
13:10-13:30	Section 2: Definitions & Preliminaries	Jiafeng Guo
13:30-14:30	Section 3: Docid design	Yubao Tang



15min coffee break

14:45-15:20	Section 4: Training approaches	Ruqing Zhang
15:20-15:40	Section 5: Inference strategies	Ruqing Zhang
15:40-16:00	Section 6: Applications	Yubao Tang
16:00-16:10	Section 7: Challenges & Opportunities	Maarten de Rijke
16:10-16:30	Q & A	All

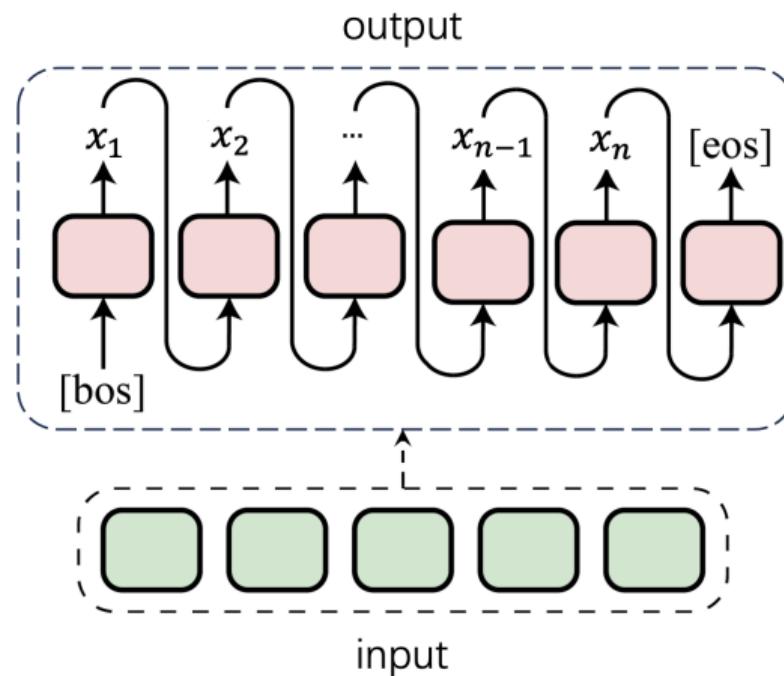
## **Section 2: Definitions & Preliminaries**

## Generative retrieval: Definition

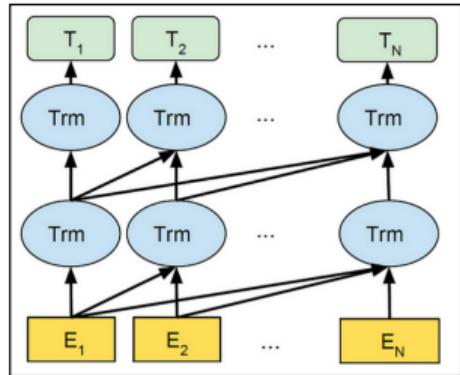
Generative retrieval (GR) aims to directly generate the **identifiers** of information resources (e.g., docids) that are relevant to an information need (e.g., an input query) **in an autoregressive fashion**

## Autoregressive formulation

$$P(x_n | x_1, x_2, \dots, x_{n-1})$$



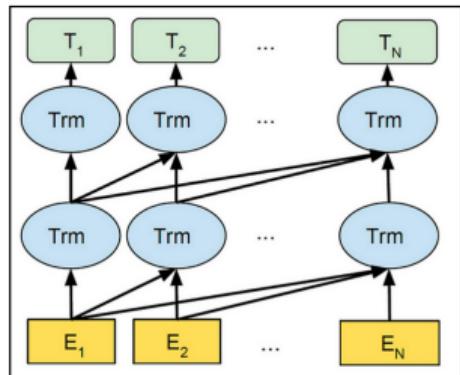
# Autoregressive models



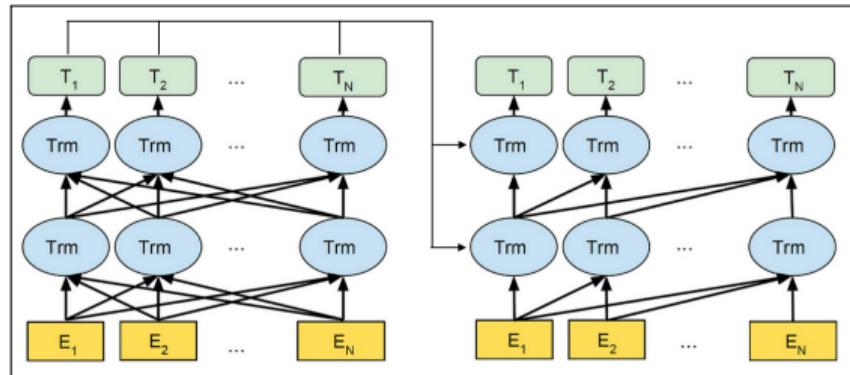
Decoder-only

Image source: [Kenton and Toutanova \[2019\]](#) and [Retrieval-based Language Models and Applications](#)

# Autoregressive models



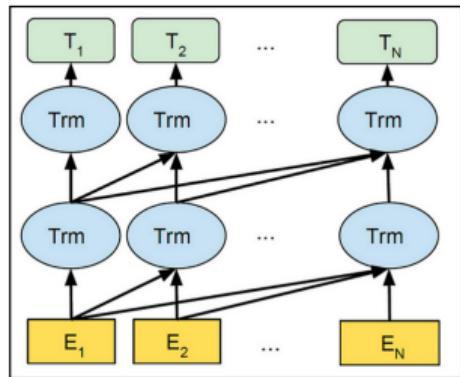
Decoder-only



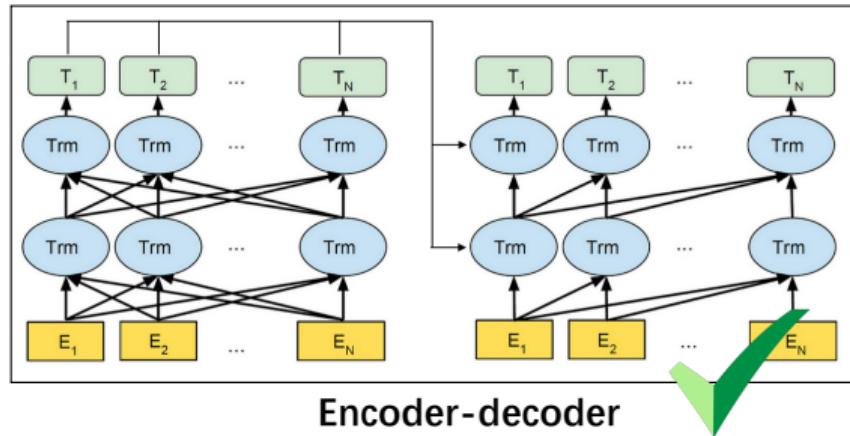
Encoder-decoder

Image source: [Kenton and Toutanova \[2019\]](#) and [Retrieval-based Language Models and Applications](#)

# Autoregressive models



Decoder-only



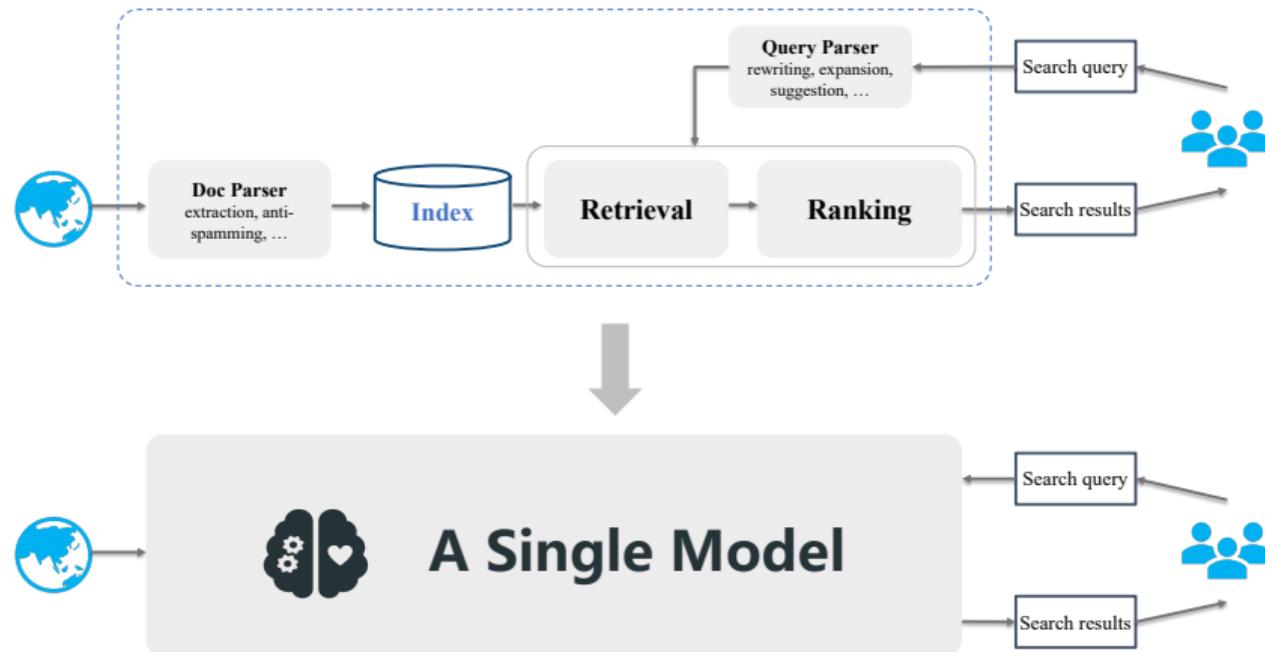
Encoder-decoder

Image source: [Kenton and Toutanova \[2019\]](#) and [Retrieval-based Language Models and Applications](#)

## Generative retrieval: Definition

GR usually exploits a Seq2Seq encoder-decoder architecture to generate a ranked list of docids for an input query, in an autoregressive fashion

## Revisit the key idea



## Two basic operations in GR

- **Indexing:** To **memorize information about each document**, a GR model should learn to associate the content of each document with its corresponding docid

## Two basic operations in GR

- **Indexing:** To **memorize information about each document**, a GR model should learn to associate the content of each document with its corresponding docid
- **Retrieval:** Given an input query, a GR model should **return a ranked list of candidate docids** by autoregressively generating the docid string

## Indexing: Formulation

Given:

- A corpus of documents  $D$ ;
- A corresponding docid set  $I_D$ ;

## Indexing: Formulation

Given:

- A corpus of **documents**  $D$ ;
- A **corresponding docid set**  $I_D$ ;

The indexing task directly takes each original document  $d \in D$  as input and generates its docid  $id \in I_D$  as output in a straightforward Seq2Seq fashion, i.e.,

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

where  $\theta$  denotes the model parameters, and  $P(id \mid d; \theta)$  is the likelihood of each docid  $id$  given the document  $d$

## Retrieval: Formulation

Given:

- A query set  $Q$ ;
- A set of relevant docids  $I_Q$ ;

## Retrieval: Formulation

Given:

- A query set  $Q$ ;
- A set of relevant docids  $I_Q$ ;

The retrieval task aims to generate a ranked list of relevant docids  $id^q \in I_Q$  in response to a query  $q \in Q$  with the indexed information, i.e.,

$$\mathcal{L}_{\text{Retrieval}}(Q, I_Q; \theta) = - \sum_{q \in Q} \sum_{id^q \in I_Q} \log P(id^q | q; \theta),$$

where  $P(id^q | q; \theta)$  is the likelihood of each relevant docid  $id^q$  given the query  $q$

## Training

Following the above two basic operations, i.e., indexing and retrieval, a single model can be optimized directly in **an end-to-end manner** towards **a global objective**,

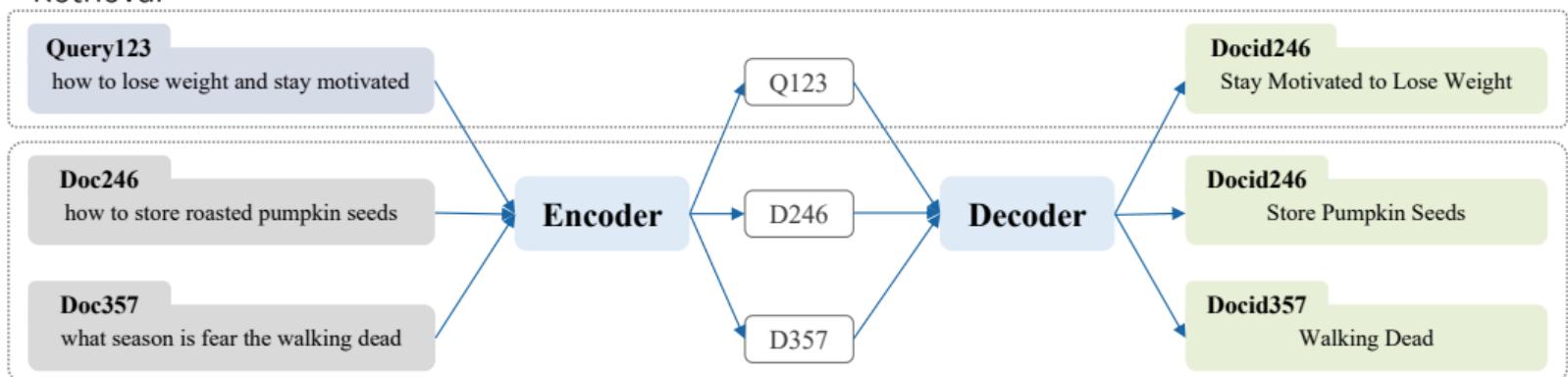
## Training

Following the above two basic operations, i.e., indexing and retrieval, a single model can be optimized directly in **an end-to-end manner** towards **a global objective**,

$$\mathcal{L}_{Global}(Q, D, I_D, I_Q; \theta) = \mathcal{L}_{Indexing}(D, I_D; \theta) + \mathcal{L}_{Retrieval}(Q, I_Q; \theta)$$

# Training: An example

Retrieval



Indexing

Joint learning the indexing and retrieval tasks

## Inference

- Once such a GR model is learned, it can be used to generate candidate docids for a test query  $q_t$ , all **within a single, unified model**,

$$w_t = GR_\theta(q_t, w_0, w_1, \dots, w_{t-1}),$$

where  $w_t$  is the  $t$ -th token in the docid string and the generation stops when decoding a special EOS token

## Inference

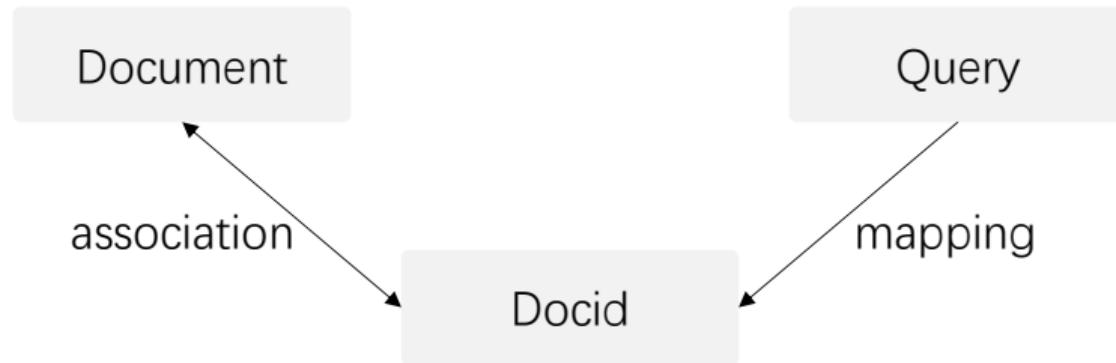
- Once such a GR model is learned, it can be used to generate candidate docids for a test query  $q_t$ , all **within a single, unified model**,

$$w_t = GR_{\theta}(q_t, w_0, w_1, \dots, w_{t-1}),$$

where  $w_t$  is the  $t$ -th token in the docid string and the generation stops when decoding a special EOS token

- The docids generated with the **top- $K$  highest** likelihood (joint probability of generated tokens within a docid) form a ranking list in descending order

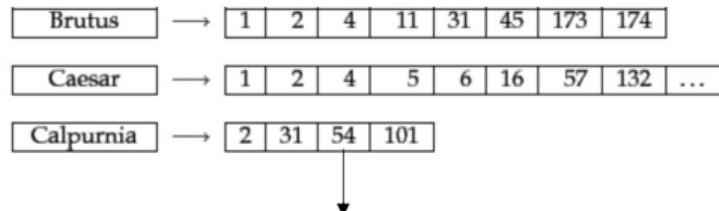
## Research questions (1): Docid design



Unfortunately, there is no natural identifier for each document!

# Research questions (1): Docid design

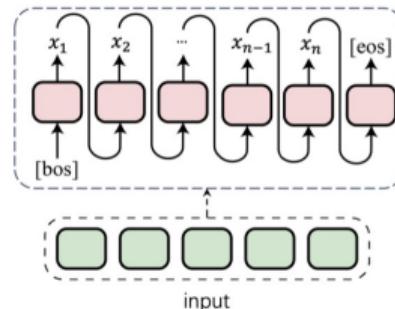
## Traditional information retrieval



As an entry

## Generative retrieval

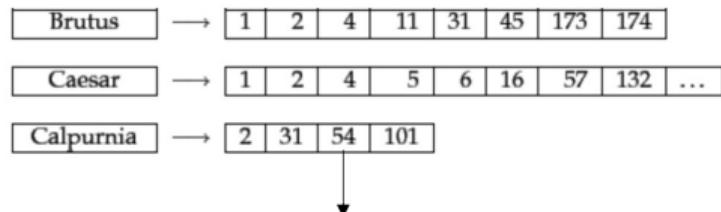
Docid: xx xxx x



For generation

# Research questions (1): Docid design

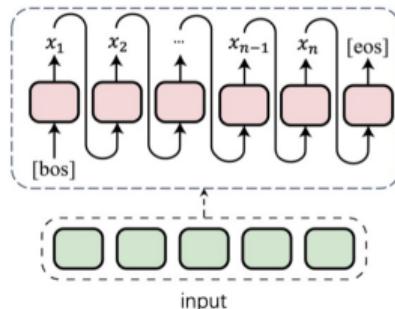
## Traditional information retrieval



As an entry

## Generative retrieval

Docid: xx xxx x



For generation

How to design docids for documents?

# Research questions (1): Docid design

- Possible design choices

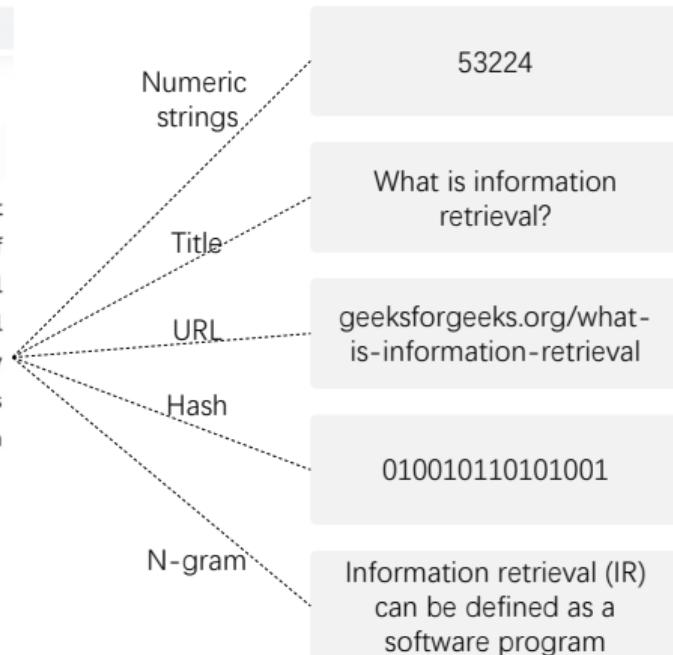
→ C geeksforgeeks.org/what-is-information-retrieval/

## What is Information Retrieval?

Read   Discuss   Courses

Information Retrieval (IR) can be defined as a software program that deals with the organization, storage, retrieval, and evaluation of information from document repositories, particularly textual information. Information Retrieval is the activity of obtaining material that can usually be documented on an unstructured nature i.e. usually text which satisfies an information need from within large collections which is stored on computers. For example, Information Retrieval can be when a user enters a query into the system.

Not only librarians, professional searchers, etc engage themselves in the activity of information retrieval but nowadays hundreds of millions of people engage in IR every day when they use web search engines. Information Retrieval is believed to be the dominant form of



## Challenges of docid design

- Shall we use randomized numbers or codes as docids?

## Challenges of docid design

- Shall we use randomized numbers or codes as docids?
- If not, how to obtain proper identifiers for documents?
  - Titles, URLs or ?

## Challenges of docid design

- Shall we use randomized numbers or codes as docids?
- If not, how to obtain proper identifiers for documents?
  - Titles, URLs or ?
- Would the choices of different docids affect the model performance (e.g., effectiveness, capacity, etc.)?
  - Long (e.g., 728 hash code) vs. Short docids (e.g., n-grams)
  - Single (e.g., title or URL) vs. Multiple docids (e.g., multiple keywords)

## Challenges of docid design

- Shall we use randomized numbers or codes as docids?
- If not, how to obtain proper identifiers for documents?
  - Titles, URLs or ?
- Would the choices of different docids affect the model performance (e.g., effectiveness, capacity, etc.)?
  - Long (e.g., 728 hash code) vs. Short docids (e.g., n-grams)
  - Single (e.g., title or URL) vs. Multiple docids (e.g., multiple keywords)

We will tackle these questions in Section 3!

## Research questions (2): Training approaches

Retrieval

Query123

how to lose weight and stay motivated

Doc246

how to store roasted pumpkin seeds

Doc357

what season is fear the walking dead

Indexer & Retriever

Docid246

Stay Motivated to Lose Weight

Docid246

Store Pumpkin Seeds

Docid357

Walking Dead

Indexing

Joint learning process of the indexing and retrieval tasks

## Challenges of training approaches

- **How to memorize the whole corpus effectively and efficiently?**
  - Rich information in documents
  - Limited labeled data

## Challenges of training approaches

- **How to memorize the whole corpus effectively and efficiently?**
  - Rich information in documents
  - Limited labeled data
- **How to learn heterogeneous tasks well within a single model?**
  - Different data distributions
  - Different optimization objectives

## Challenges of training approaches

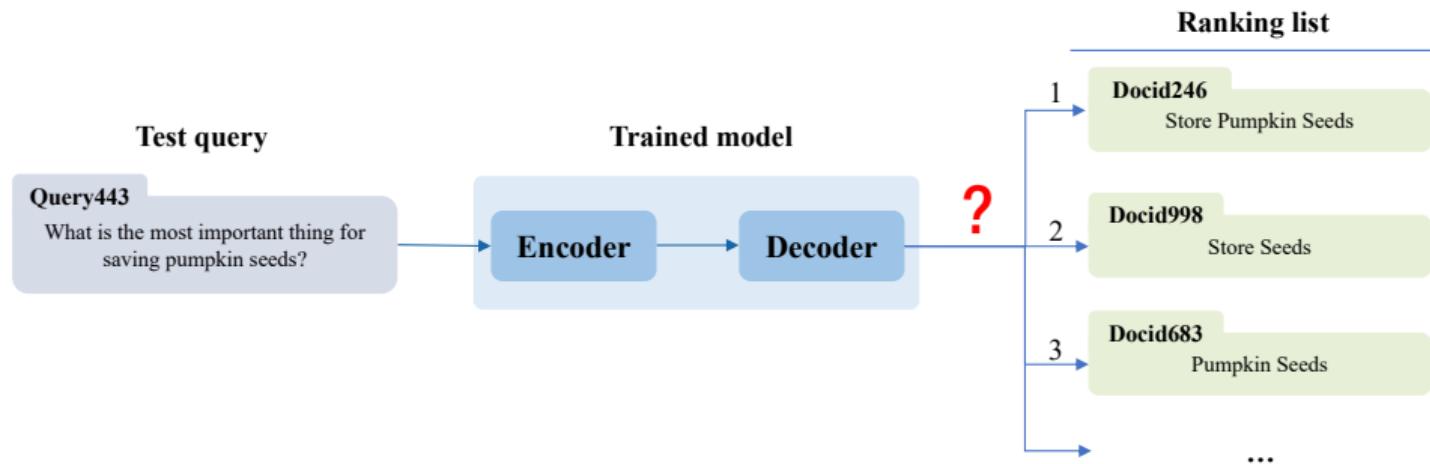
- **How to memorize the whole corpus effectively and efficiently?**
  - Rich information in documents
  - Limited labeled data
- **How to learn heterogeneous tasks well within a single model?**
  - Different data distributions
  - Different optimization objectives
- **How to handle a dynamically evolving document collection?**
  - Internal index: model parameters
  - High computational costs: re-training from scratch every time the underlying corpus is updated

## Challenges of training approaches

- **How to memorize the whole corpus effectively and efficiently?**
  - Rich information in documents
  - Limited labeled data
- **How to learn heterogeneous tasks well within a single model?**
  - Different data distributions
  - Different optimization objectives
- **How to handle a dynamically evolving document collection?**
  - Internal index: model parameters
  - High computational costs: re-training from scratch every time the underlying corpus is updated

We will tackle these questions in Section 4!

## Research questions (3): Inference strategies



The generation process is different from general language generation

## Challenges of model inference

- **How to generate valid docids?**
  - Limited docids vs. free generation

## Challenges of model inference

- **How to generate valid docids?**
  - Limited docids vs. free generation
- **How to organize the docids for large scale corpus?**
  - Data structure for docids over millions of documents

## Challenges of model inference

- **How to generate valid docids?**
  - Limited docids vs. free generation
- **How to organize the docids for large scale corpus?**
  - Data structure for docids over millions of documents
- **How to generate a ranked list of docids for a query?**
  - One-by-one generation: likelihood probabilities
  - One-time generation: directly decoding a sequence of docids

## Challenges of model inference

- **How to generate valid docids?**
  - Limited docids vs. free generation
- **How to organize the docids for large scale corpus?**
  - Data structure for docids over millions of documents
- **How to generate a ranked list of docids for a query?**
  - One-by-one generation: likelihood probabilities
  - One-time generation: directly decoding a sequence of docids

We will tackle these questions in Section 5!

## Research questions (4): Applications

How to employ generative retrieval models in different downstream tasks?

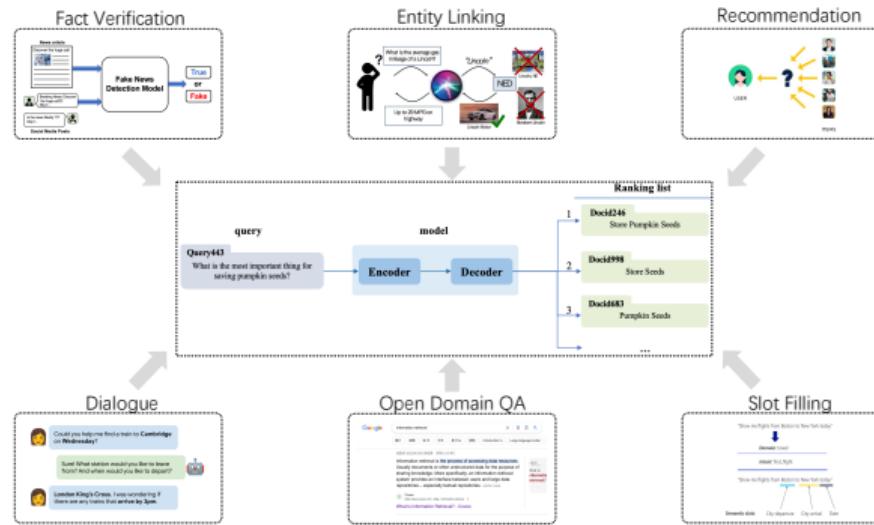
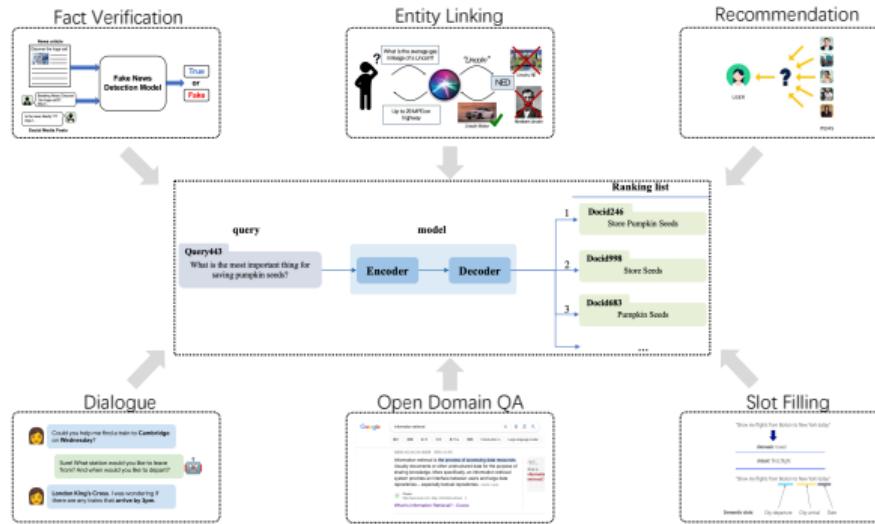


Image source: Murayama [2021], Stanford blog, Froomle AI, Google, Zero-shot blog, and Heck and Heck [2020]

## Research questions (4): Applications

How to employ generative retrieval models in different downstream tasks?



We will tackle this question in Section 6!

Image source: Murayama [2021], Stanford blog, Froomle AI, Google, Zero-shot blog, and Heck and Heck [2020]

## **Section 3: Docid design**

## Challenges of docid design

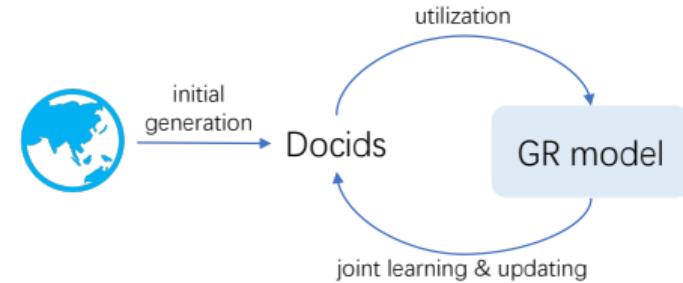
- Shall we use randomize numbers as the docids?
- If not, how to construct proper docids for the documents?
- Would the choices of different docids affect the model performance (effectiveness, capacity, etc.)?

## Categorization of docids



- Pre-defined static docids

## Categorization of docids



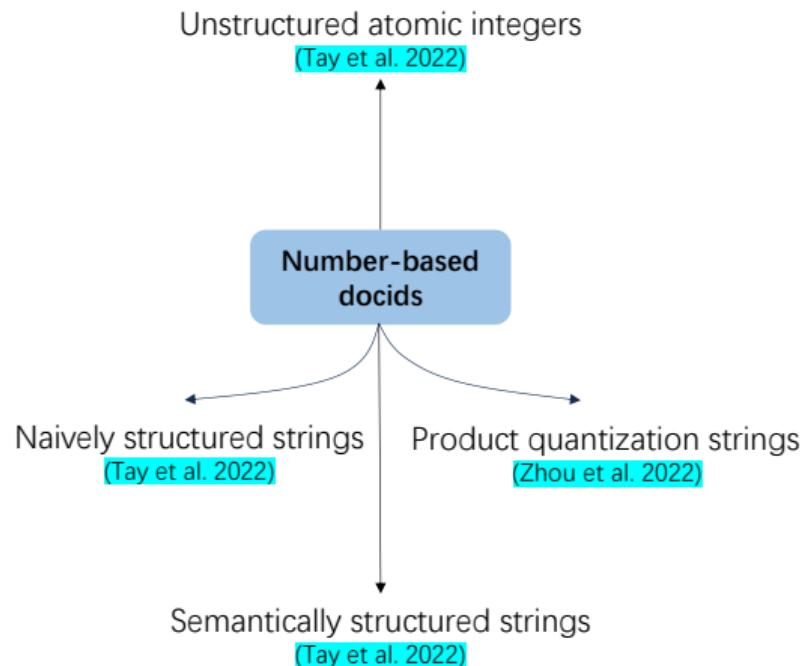
- Pre-defined static docids

- Learnable docids

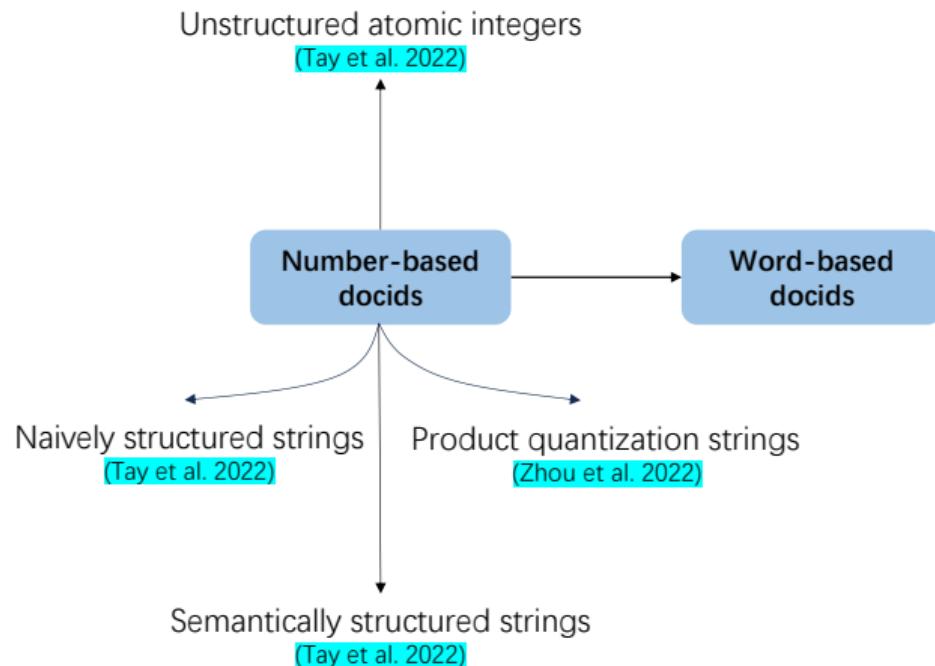
# Roadmap of pre-defined static docids

Number-based  
docids

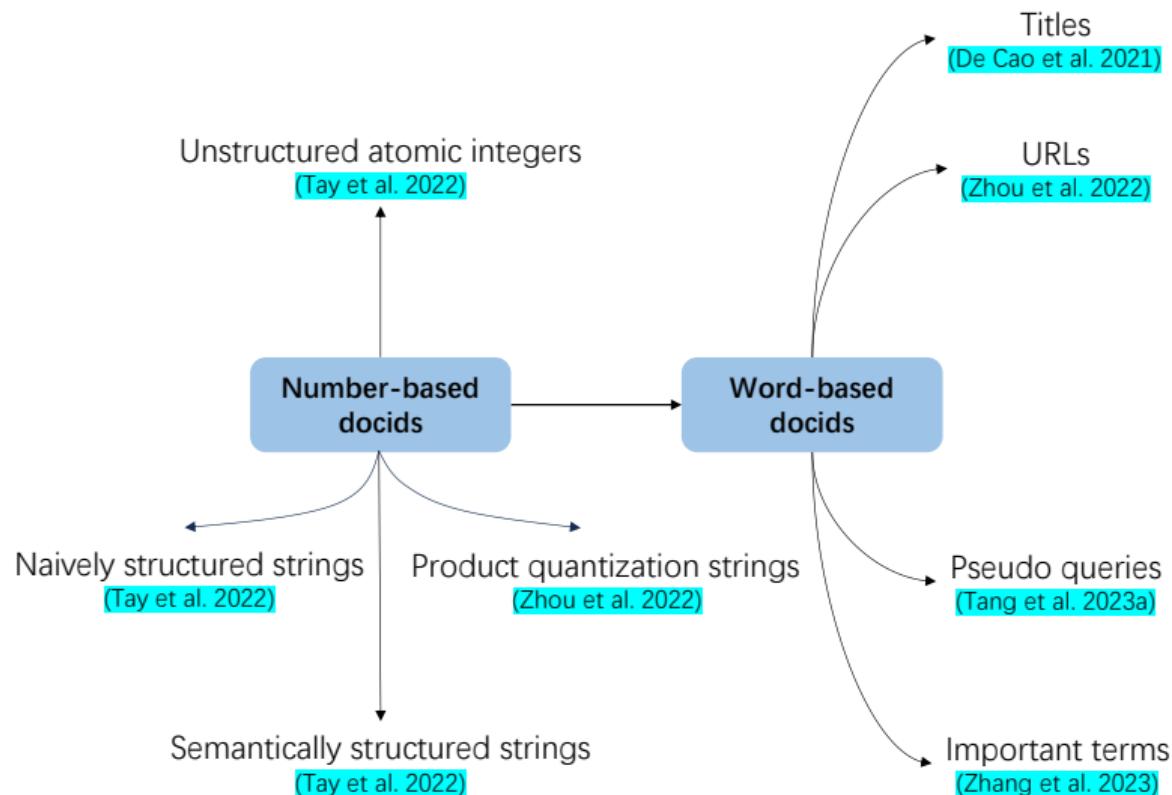
# Roadmap of pre-defined static docids



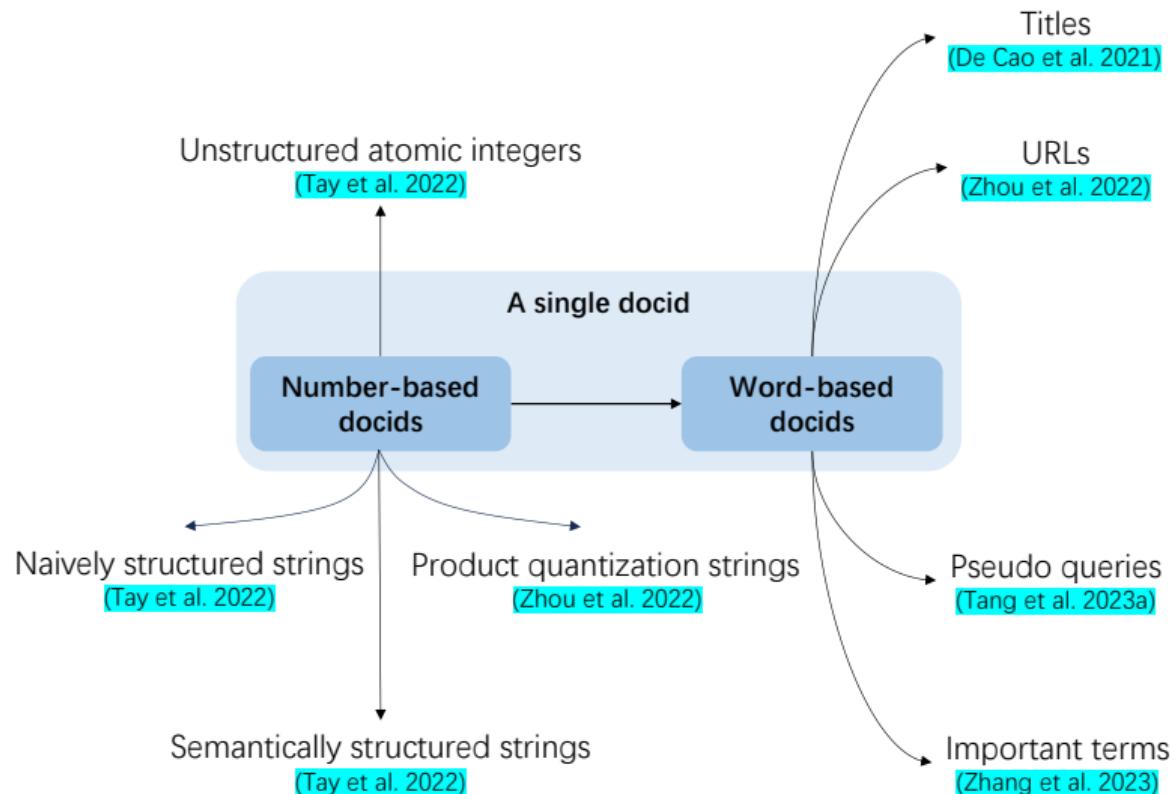
# Roadmap of pre-defined static docids



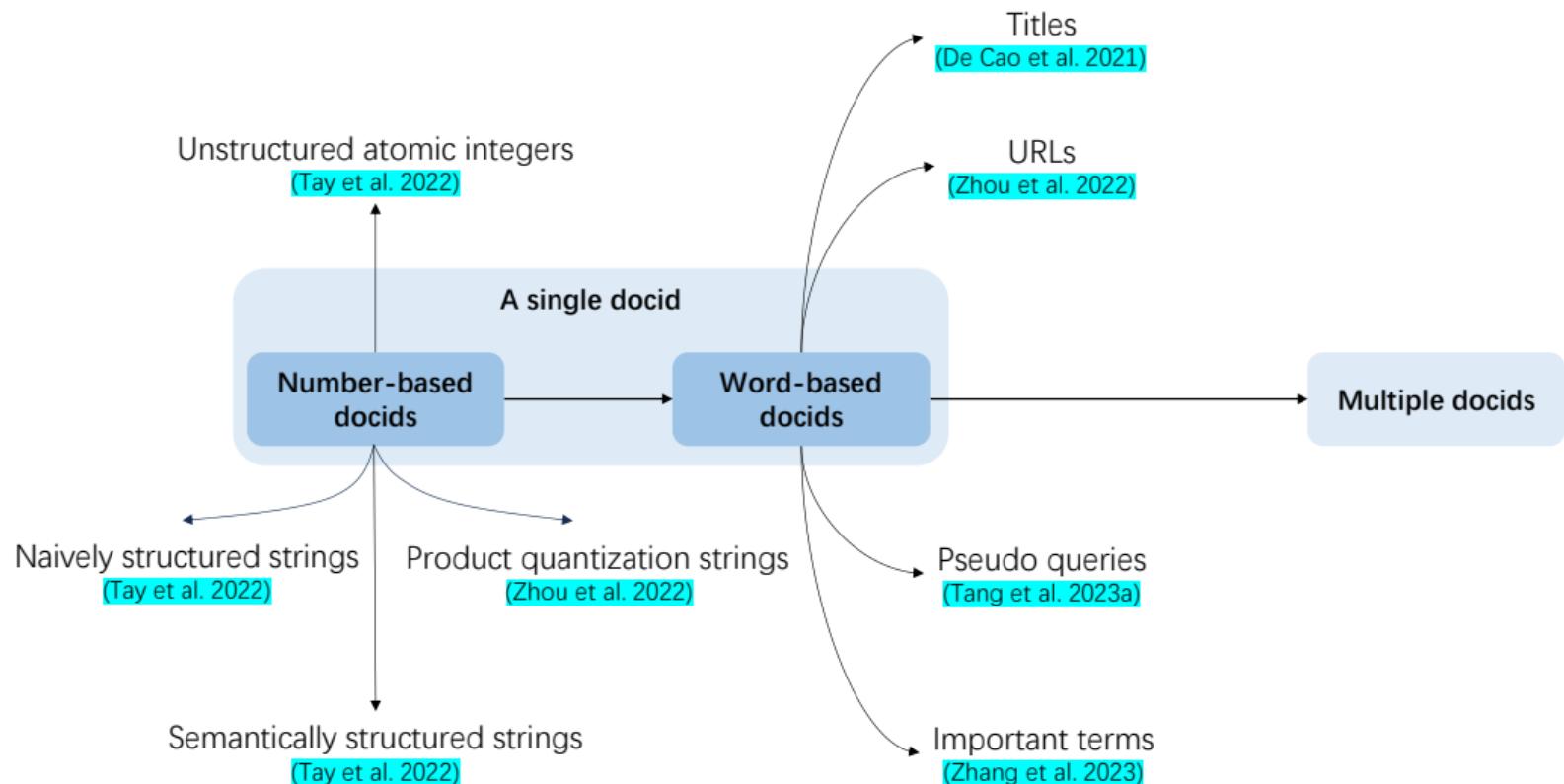
# Roadmap of pre-defined static docids



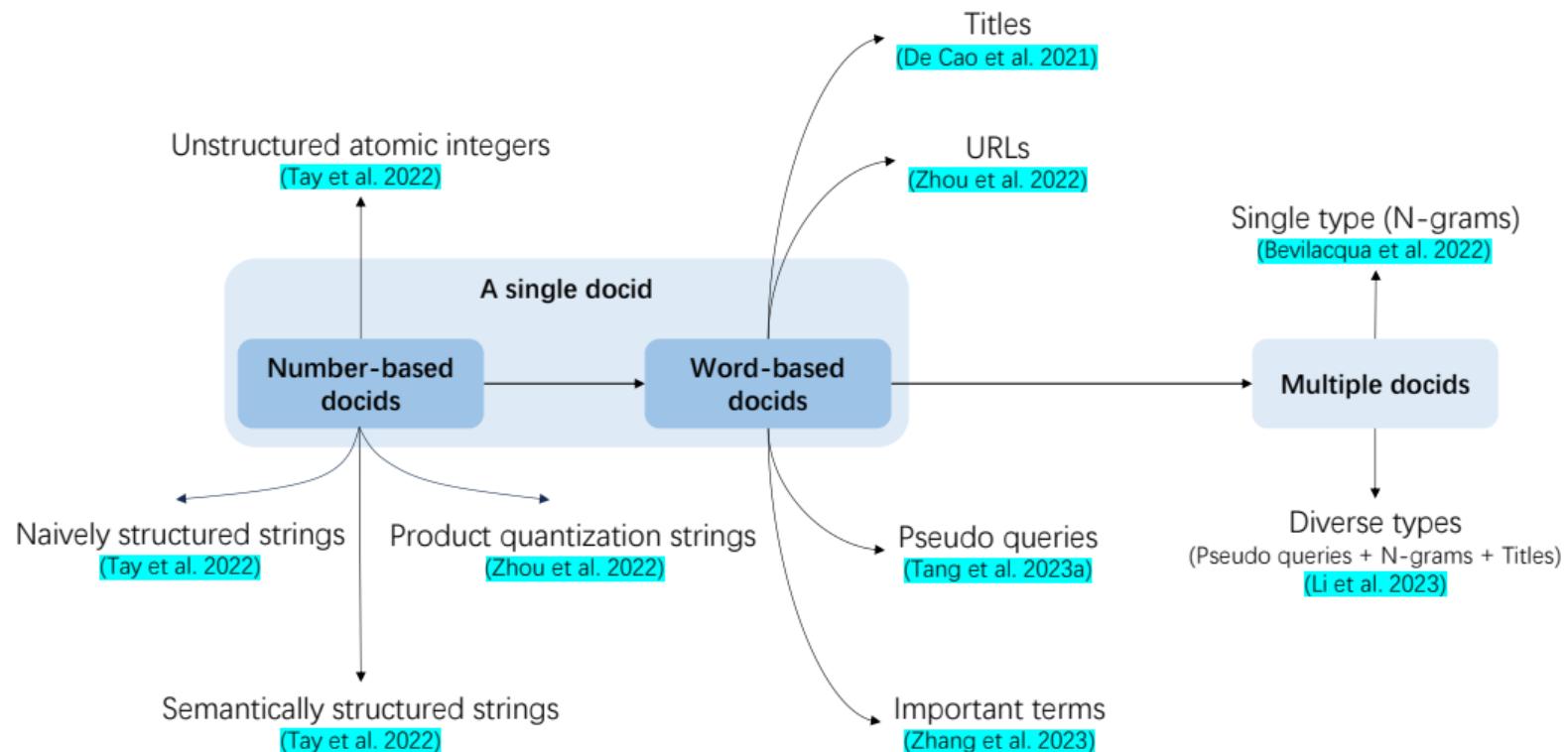
# Roadmap of pre-defined static docids



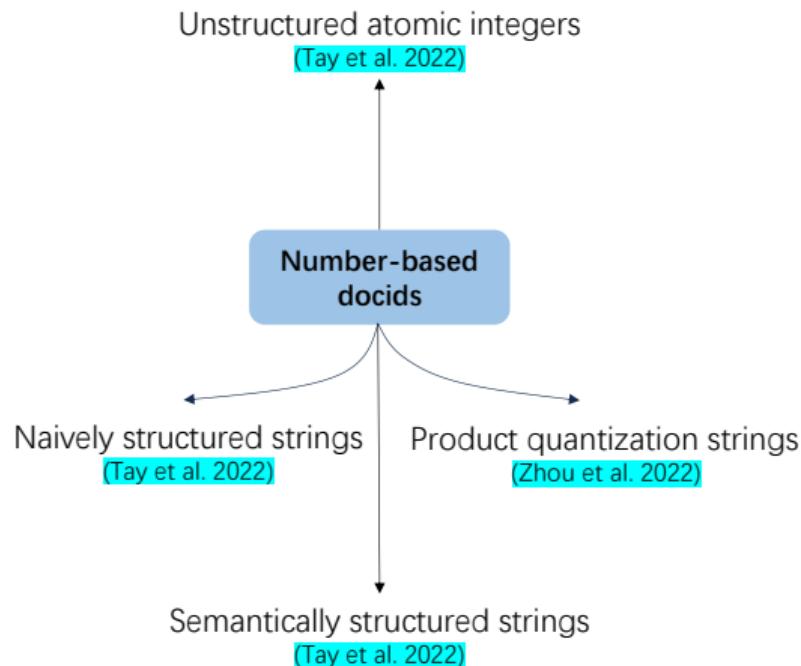
# Roadmap of pre-defined static docids



# Roadmap of pre-defined static docids



## A single docid: Number-based



## Number-based: Unstructured atomic integers

- An arbitrary (and possibly random) unique integer identifier

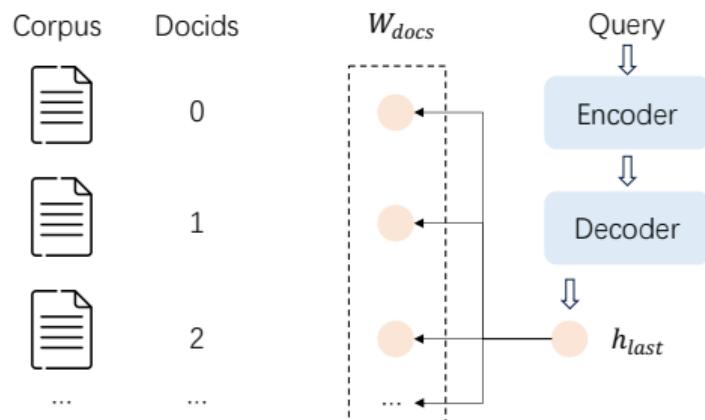
## Number-based: Unstructured atomic integers

- An arbitrary (and possibly random) unique integer identifier

Corpus	Docids
	0
	1
	2
...	...

## Number-based: Unstructured atomic integers

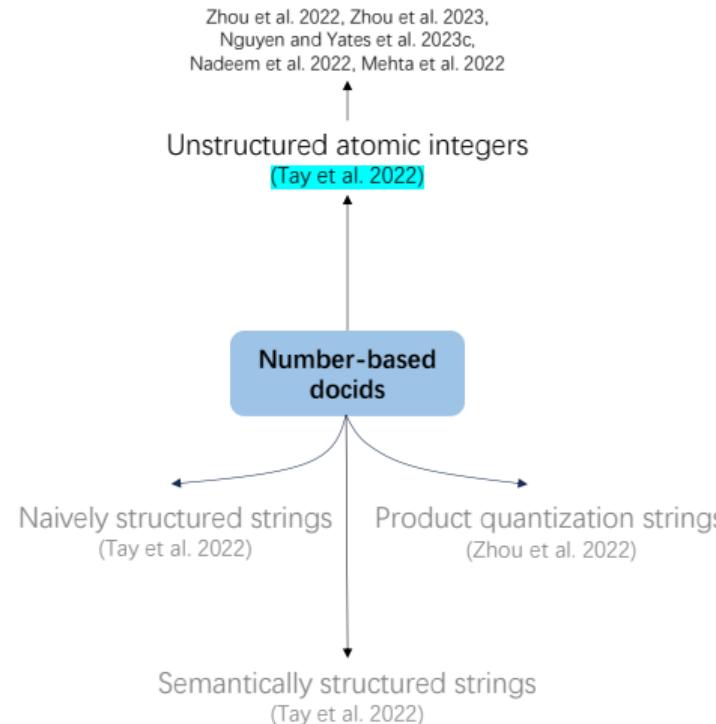
- **Decoding formulation:** learn a probability distribution over the docid embeddings, i.e., emitting one logit for each unique docid



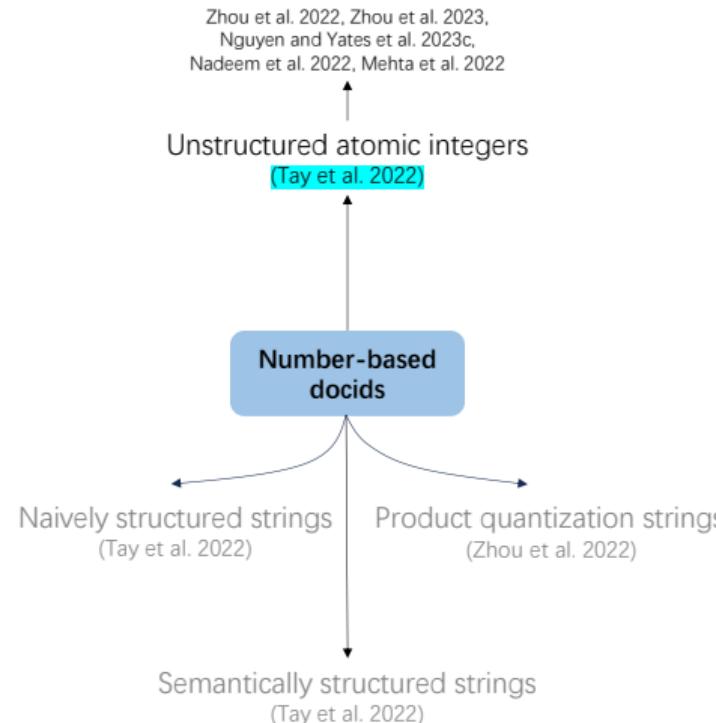
$$O = \text{Softmax}([W_{docs}]^T h_{last}),$$

where  $[W_{docs}]$  is the document embedding matrix, and  $h_{last}$  is the last layer's hidden state of the decoder

# Unstructured atomic integers and subsequent work



# Unstructured atomic integers and subsequent work



Easy to build: analogous to the output layer in standard language model

## Unstructured atomic integers: obvious constraints



The need to learn embeddings for each individual document

## Unstructured atomic integers: obvious constraints



The need to learn embeddings for each individual document



The need for the large softmax output space

## Unstructured atomic integers: obvious constraints



The need to learn embeddings for each individual document

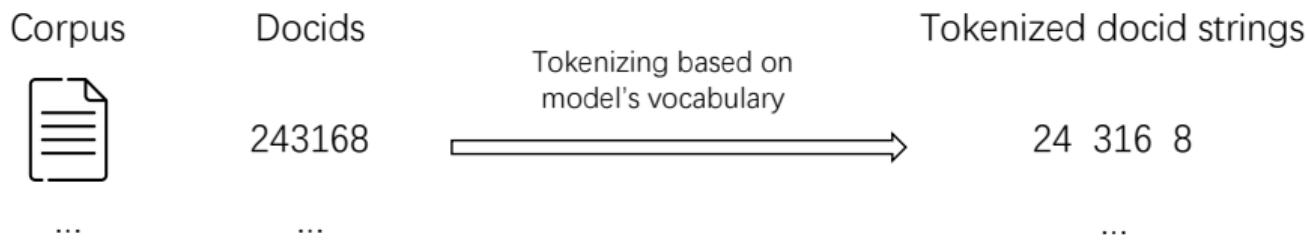


The need for the large softmax output space

**It is challenging to be used on large corpora!**

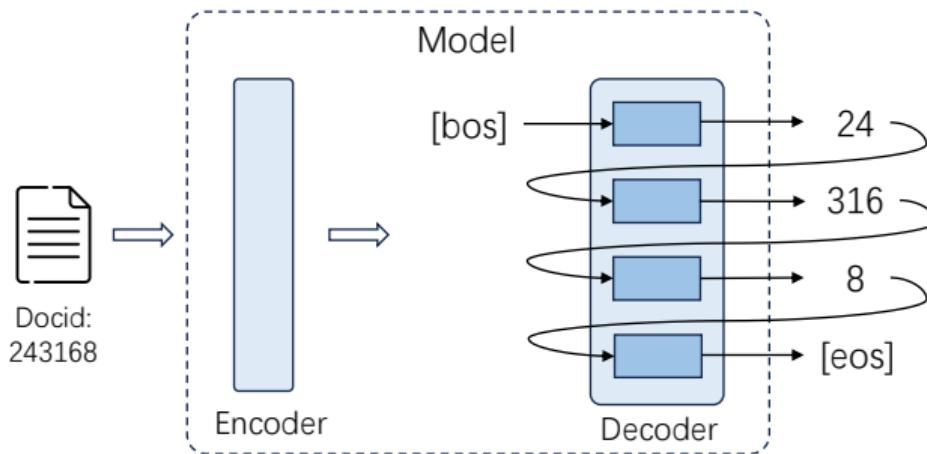
## Number-based: Naively structured strings

- Treat arbitrary unique integers as tokenizable strings

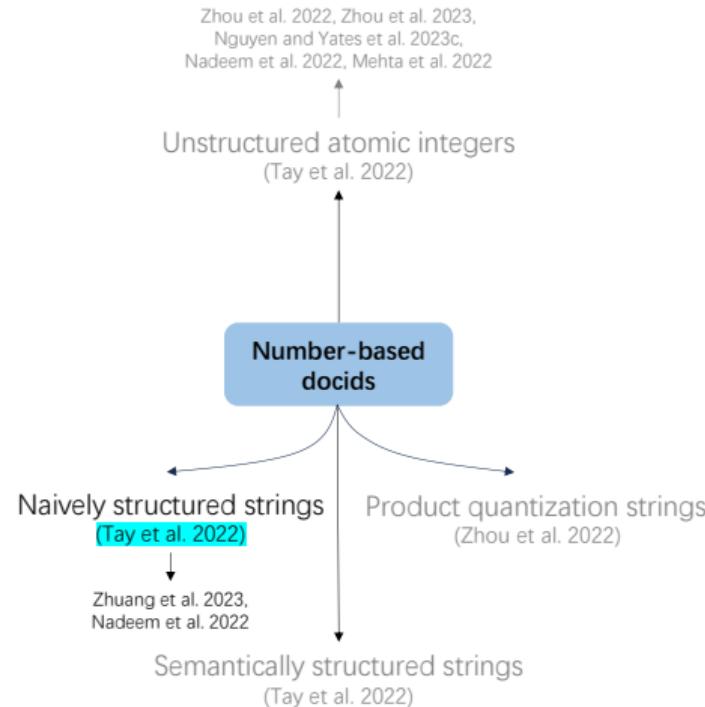


## Number-based: Naively structured strings

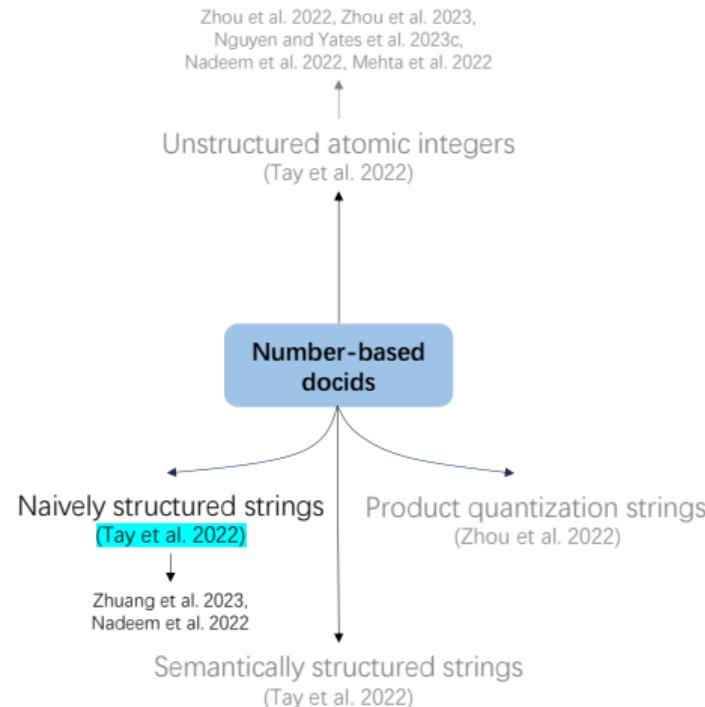
- **Decoding formulation:** Generating a docid string in a token-by-token manner



# Naively structured strings and subsequent work



# Naively structured strings and subsequent work



Such a way frees the limitation for the **corpus size** that comes with unstructured atomic docid

## Naively structured strings: obvious constraints



Identifiers are assigned in an **arbitrary manner**

## Naively structured strings: obvious constraints



Identifiers are assigned in an **arbitrary manner**



The docid space **lacks semantic structure**

## Number-based: Semantically structured strings

Properties:

- The docid should capture **some information about the semantics** of its associated document

## Number-based: Semantically structured strings

Properties:

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

## Number-based: Semantically structured strings

Properties:

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



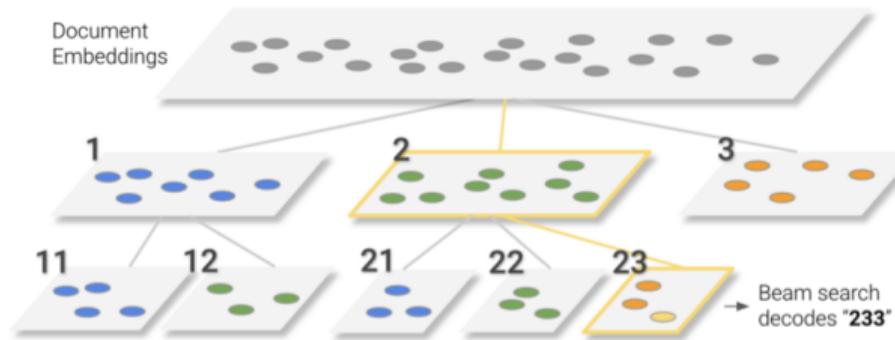
Semantically similar documents share identifier prefixes

## Number-based: Semantically structured strings

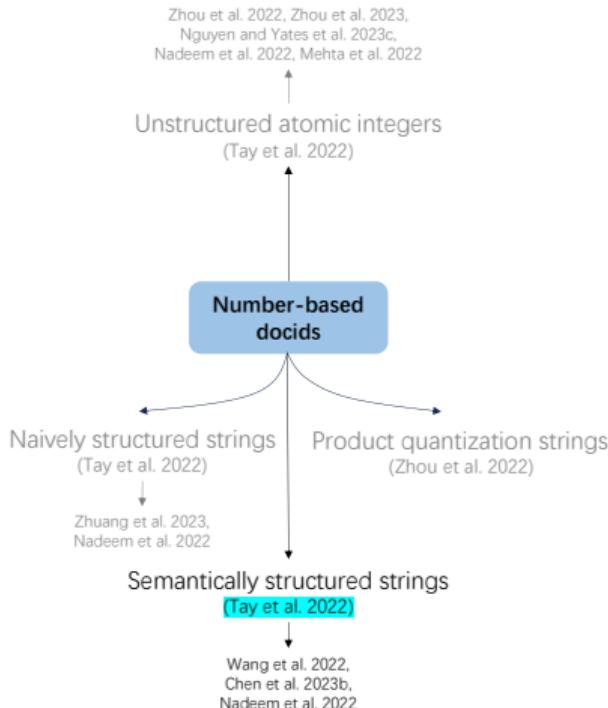
- A hierarchical clustering algorithm over document embeddings to induce a decimal tree

## Number-based: Semantically structured strings

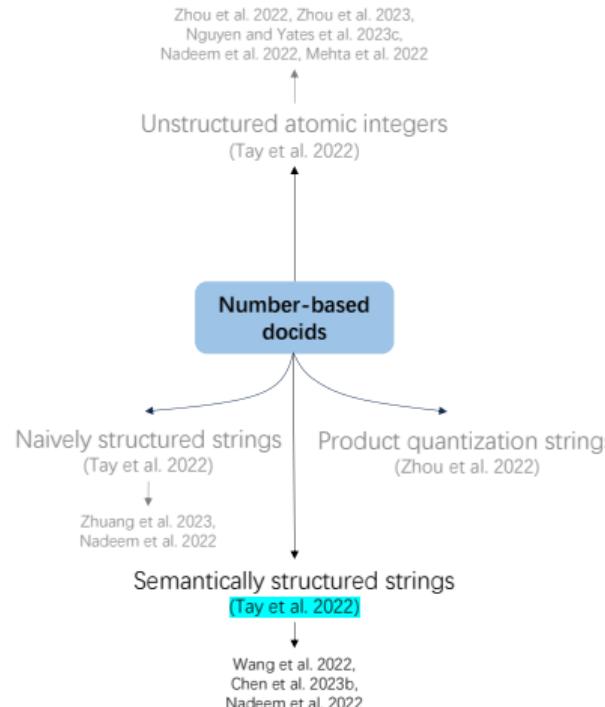
- A hierarchical clustering algorithm over document embeddings to induce a decimal tree



# Semantically structured strings and subsequent work

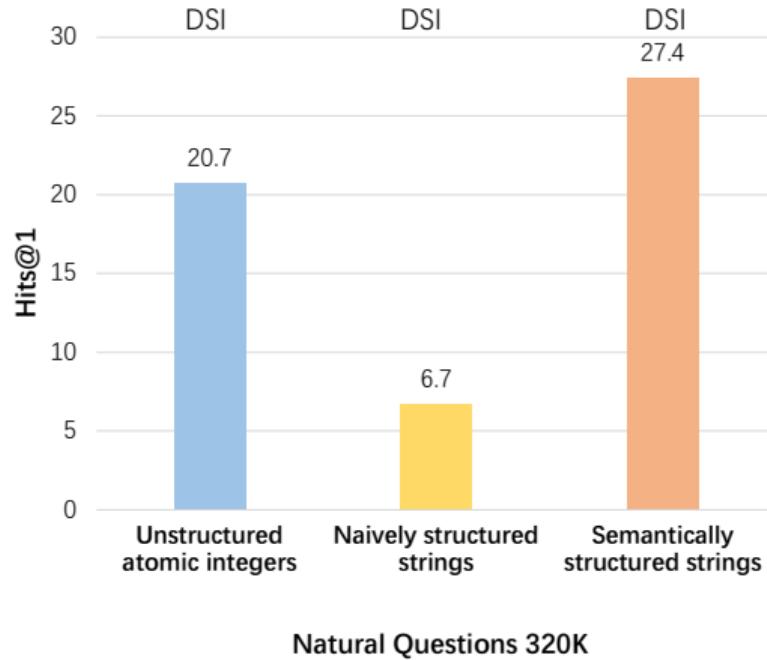


# Semantically structured strings and subsequent work



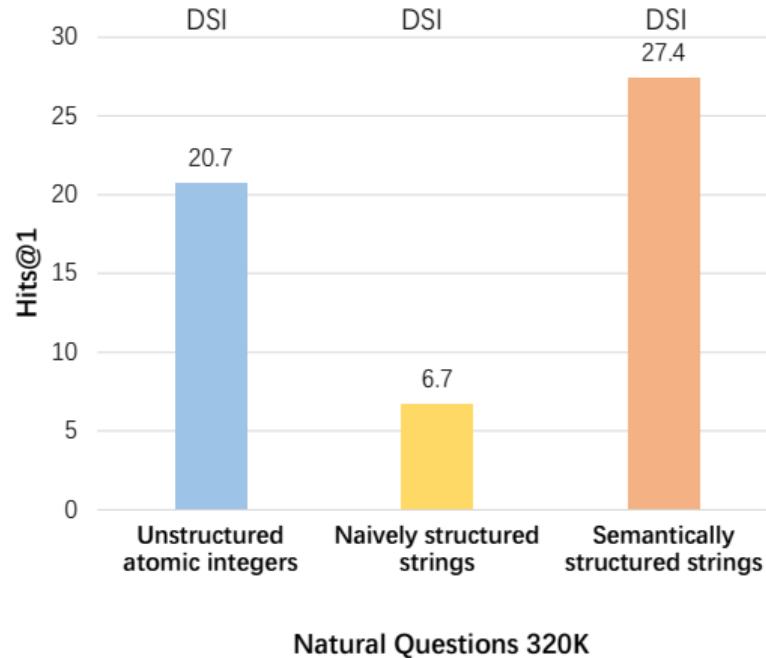
- The document semantics can be incorporated in the decoding process
- It is not limited by the size of the corpus

## Performance comparisons [Tay et al., 2022]



- Backbone: T5-base
- Observations: imbuing the docid space with semantic structure can lead to better retrieval capabilities

## Performance comparisons [Tay et al., 2022]



- Backbone: T5-base
- Observations: imbuing the docid space with semantic structure can lead to better retrieval capabilities

This is only about "identifiers"

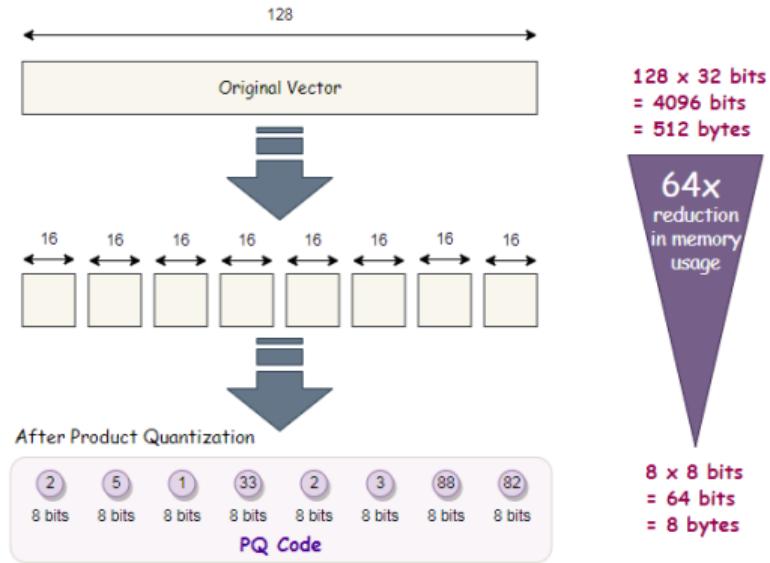
Later sections will discuss the performance compared to traditional IR models

## Number-based: Product quantization strings

- Product quantization (PQ) is a technique used for vector compression

## Number-based: Product quantization strings

- Product quantization (PQ) is a technique used for vector compression
- An original vector is represented by a short code composed of its subspace quantization indices



## Number-based: Product quantization strings

Given all  $D$ -dimensional embedding vectors of documents [Zhou et al., 2022],

## Number-based: Product quantization strings

Given all  $D$ -dimensional embedding vectors of documents [Zhou et al., 2022],

- Divide the  $D$ -dimensional space into  $m$  groups

## Number-based: Product quantization strings

Given all  $D$ -dimensional embedding vectors of documents [Zhou et al., 2022],

- Divide the  $D$ -dimensional space into  $m$  groups
- Perform  $K$ -means clustering on each group to obtain  $k$  cluster centers

## Number-based: Product quantization strings

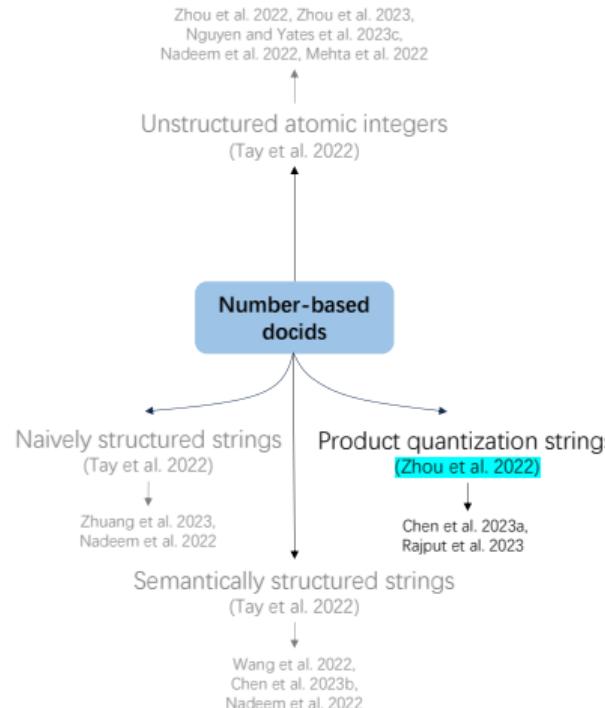
Given all  $D$ -dimensional embedding vectors of documents [Zhou et al., 2022],

- Divide the  $D$ -dimensional space into  $m$  groups
- Perform  $K$ -means clustering on each group to obtain  $k$  cluster centers
- Each embedding vector can be represented as a set of  $m$  cluster identifiers. For each document  $d$ , its product quantization string identifier  $id_{PQ}$  can be defined,

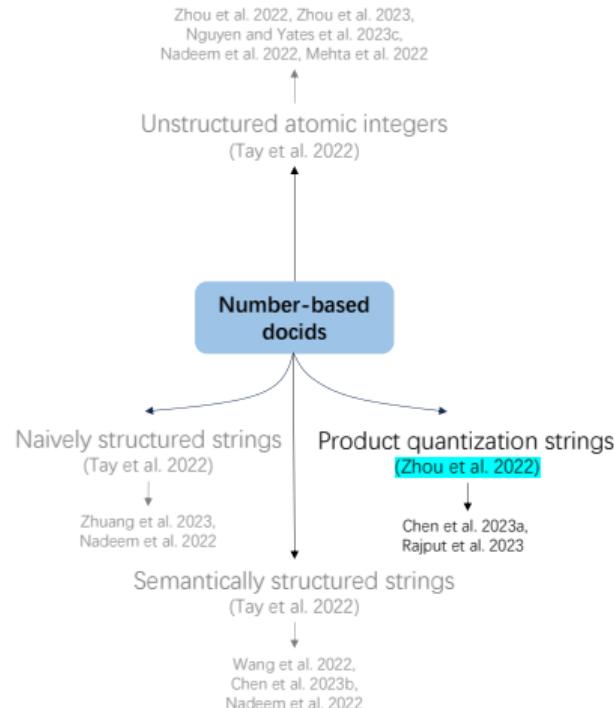
$$id_{PQ} = PQ(Encoder(d)),$$

where  $Encoder(\cdot)$  can be implemented by different language models

# Product quantization strings and subsequent work



# Product quantization strings and subsequent work

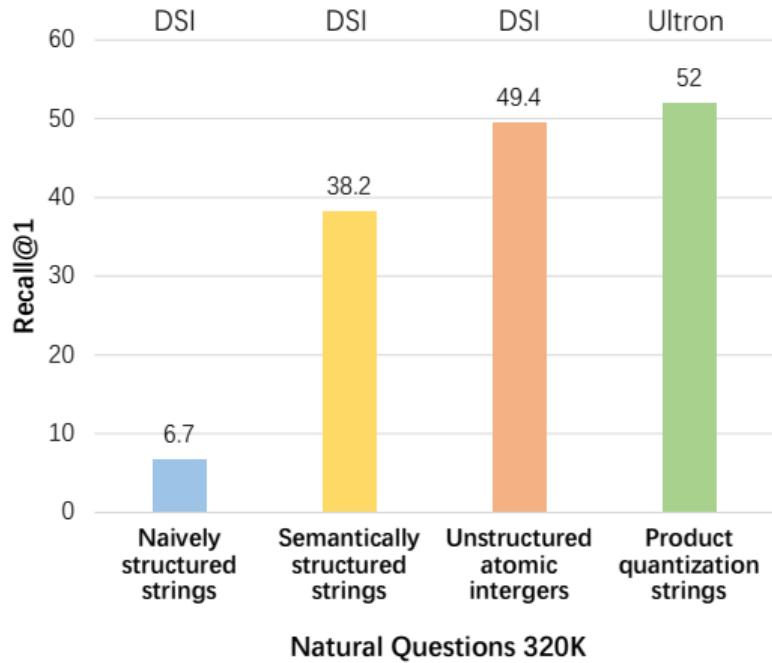


Preserving dense vector semantics in a smaller space



Capturing local semantic information

# Performance comparisons



- Backbone: T5-base
- Observations: Product quantization string identifiers improves over structured semantic identifiers

## Number-based docids: Summary



Docids based on integers are easy to build

## Number-based docids: Summary

- thumb up Docids based on integers are easy to build
- thumb up Unstructured atomic integers and naively/semantically structured strings can maintain **uniqueness**

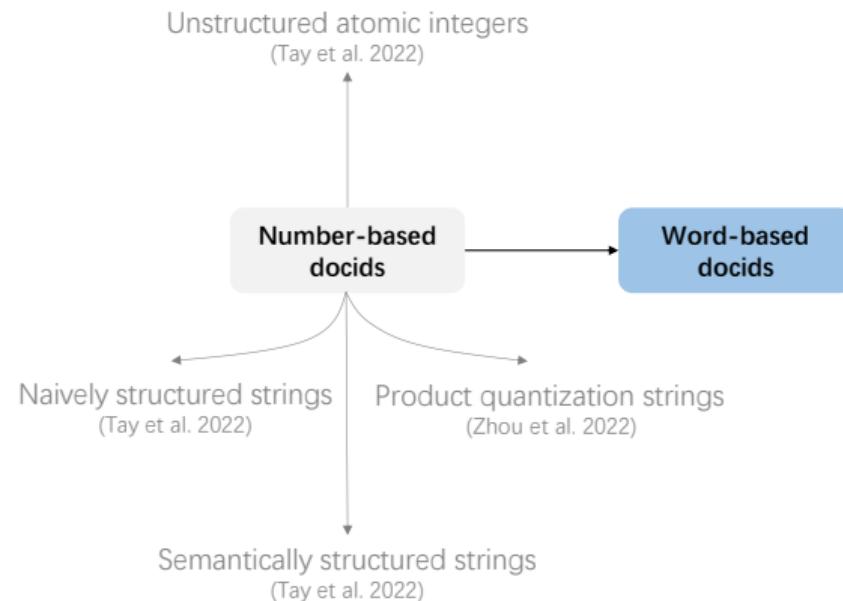
## Number-based docids: Summary

-  Docids based on integers are easy to build
-  Unstructured atomic integers and naively/semantically structured strings can maintain **uniqueness**
-  Number-based docids are composed of **unreadable numbers**

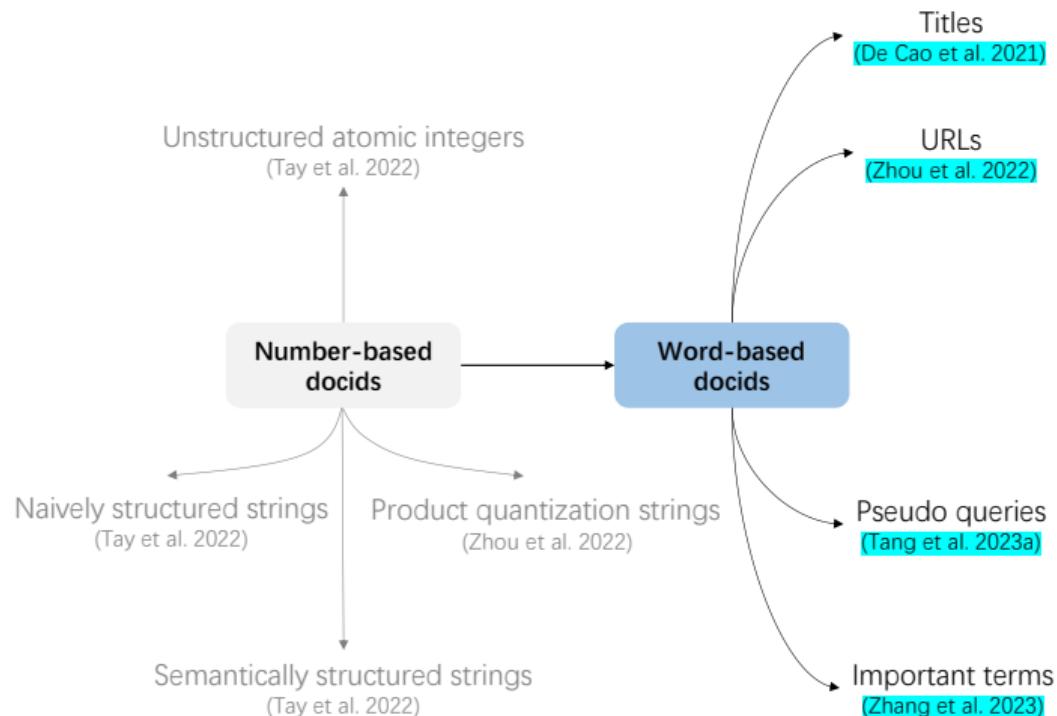
## Number-based docids: Summary

-  Docids based on integers are easy to build
-  Unstructured atomic integers and naively/semantically structured strings can maintain **uniqueness**
-  Number-based docids are composed of **unreadable numbers**
-  It is challenging to **interpret** the model's understanding of the corpus

## A single docid: Word-based



# A single docid: Word-based



## A single docid: Word-based

The fundamental inspiration

- The query is usually keyword-based **natural language**, which can be challenging to map into a **numeric string**, while mapping it to words would be more intuitive

## A single docid: Word-based

The fundamental inspiration

- The query is usually keyword-based **natural language**, which can be challenging to map into a **numeric string**, while mapping it to words would be more intuitive
- Elaboration Strategies in human learning encoding and recall for humans: natural language vs. integer-based strings

## Word-based: Titles

- Document titles: be able to summarize the main content

# Word-based: Titles

- Document titles: be able to summarize the main content

**Information retrieval** Decoding target

Article Talk

From Wikipedia, the free encyclopedia

**Information retrieval (IR)** in computing and information science is the process of obtaining information system resources that are relevant to an information need from a collection of those resources. Searches can be based on full-text or other content-based indexing. Information retrieval is the science<sup>[1]</sup> of searching for information in a document, searching for documents themselves, and also searching for the metadata that describes data, and for databases of texts, images or sounds.

Automated information retrieval systems are used to reduce what has been called information overload. An IR system is a software system that provides access to books, journals and other documents; it also stores and manages those documents. Web search engines are the most visible IR applications.

**Chiamaka Nnadozie's father didn't want her to play soccer. Nigerian star defied him and rewrote the record books** Decoding target

By Michael Johnston and Amanda Davies, CNN

5 minute read · Updated 10:06 AM EDT, Wed November 1, 2023

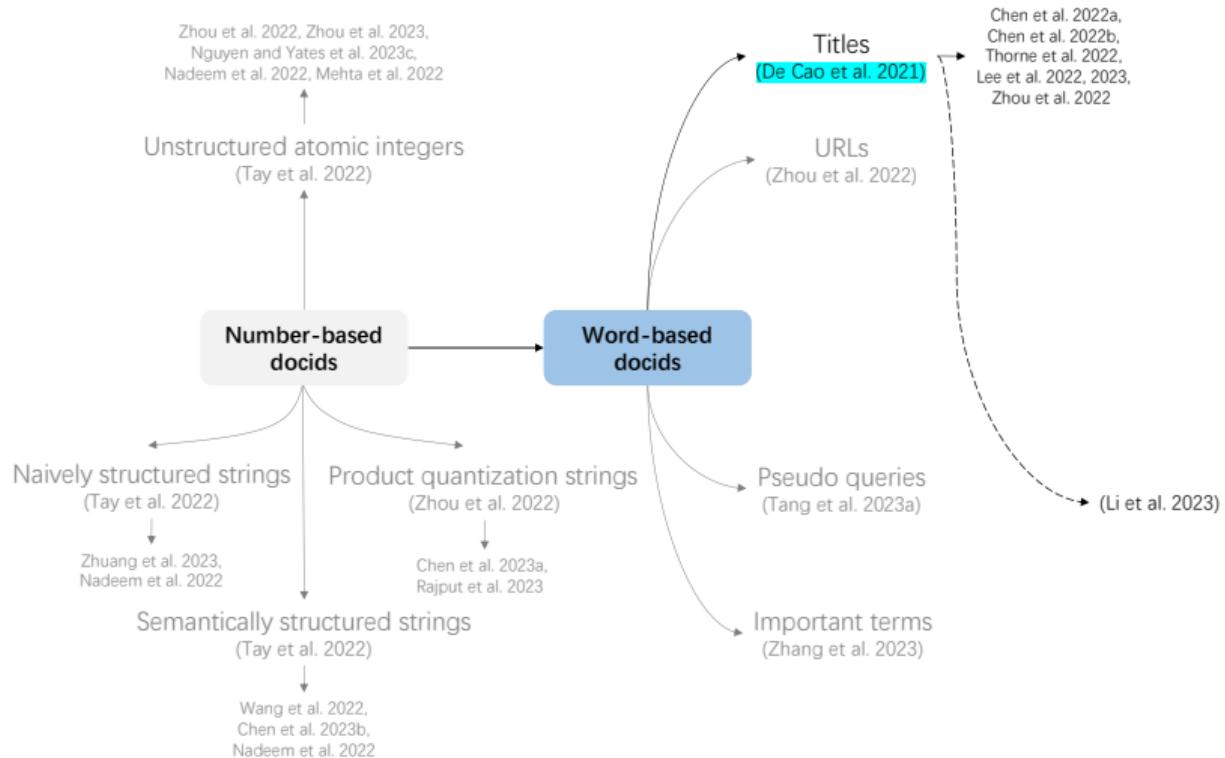
**(CNN) —** It wasn't always plain sailing for Paris FC and Nigerian goalkeeper, Chiamaka Nnadozie, throughout her now-flourishing career.

Growing up in a family of boys and men – who had all tried their hand at going professional – Nnadozie's ambition to follow suit wasn't greeted with unyielding enthusiasm. Quite the opposite.

"It wasn't very good from my family. They never let me play, especially my dad," the 22-year-old told CNN's Amanda Davies.

"Whenever I went to play soccer, he would always tell me: 'Girls don't play football. Look at me. I played football, I didn't make it. Your brother, he played, he didn't make Your cousin played, he didn't make it. So why do you want to choose this? Why don't you want to go to school or maybe do some other things?'" Nnadozie recollects.

# Titles and subsequent work



## **Titles: Obvious constraints**



Depending on certain special document metadata

## Titles: Obvious constraints

-  Depending on certain special document metadata
-  The titles may be duplicated (i.e., web datasets), and require further investigation

## Titles: Obvious constraints

-  Depending on certain special document metadata
-  The titles may be duplicated (i.e., web datasets), and require further investigation
-  Time-consuming step of producing titles and requiring increasingly sophisticated domain knowledge

For a while, mainly evaluated on Wikipedia-based tasks (with well-written titles)!

# Wikipedia-based tasks

## Fact Verification

De Cao et al. 2021, Chen et al. 2022b,  
Chen et al. 2022a, Thorne et al. 2022,  
Lee et al. 2023

## Entity Linking

De Cao et al. 2021, Chen et al. 2022b,  
Lee et al. 2023

## Slot Filling

De Cao et al. 2021, Chen et al. 2022b,  
Lee et al. 2023

## Open Domain QA

De Cao et al. 2021, Chen et al. 2022b,  
Zhou et al. 2022, Lee et al. 2023

## Dialogue

De Cao et al. 2021, Chen et al. 2022b,  
Lee et al. 2023

## Multi-hop retrieval

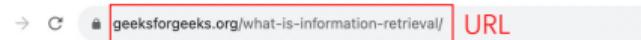
Lee et al. 2022

## Word-based: URLs

- The URL of a document contains certain semantic information and can uniquely correspond to this document

## Word-based: URLs

- The URL of a document contains certain semantic information and can uniquely correspond to this document



The screenshot shows a web browser window with the URL bar containing "geeksforgeeks.org/what-is-information-retrieval/" highlighted by a red box. The main content area displays the title "What is Information Retrieval?" and a detailed paragraph about the definition of Information Retrieval (IR) as a software program that deals with organization, storage, retrieval, and evaluation of information from document repositories. Below this, there is another paragraph about the activity of information retrieval and its prevalence in modern search engines.

What is Information Retrieval?

Information Retrieval (IR) can be defined as a software program that deals with the organization, storage, retrieval, and evaluation of information from document repositories, particularly textual information. Information Retrieval is the activity of obtaining material that can usually be documented on an unstructured nature i.e. usually text which satisfies an information need from within large collections which is stored on computers. For example, Information Retrieval can be when a user enters a query into the system.

Not only librarians, professional searchers, etc engage themselves in the activity of information retrieval but nowadays hundreds of millions of people engage in IR every day when they use web search engines. Information Retrieval is believed to be the dominant form of

## Word-based: URLs

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



tokenize

`https :// en . Wikipedia . org / wiki / N e vada`

- Ren et al. [2023] solely utilized tokenized URLs as the identifier

## Word-based: URLs

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



tokenize

`https :// en . Wikipedia . org / wiki / N e vada`

- Ren et al. [2023] solely utilized tokenized URLs as the identifier
- The tokenized symbols of URLs are well aligned with the vocabulary of the generative language model, thereby enhancing the generative capacity

## Word-based: URLs

- However, not all URLs provide sufficient semantic information

## Word-based: URLs

- However, not all URLs provide sufficient semantic information
- Zhou et al. [2022] proposed to combine the URL and the document title as docids to guarantee both the uniqueness and semantics of the identifiers

## Word-based: URLs

- However, not all URLs provide sufficient semantic information
- Zhou et al. [2022] proposed to combine the URL and the document title as docids to guarantee both the uniqueness and semantics of the identifiers

For a while, mainly evaluated on Web search datasets (with available URLs)!

# Web search datasets

## MS MARCO

Nguyen et al. 2016

## Natural Questions

Kwiatkowski et al. 2019

## Trec-CAR

Dietz et al. 2017

## Robust04

Voorhees et al. 2004

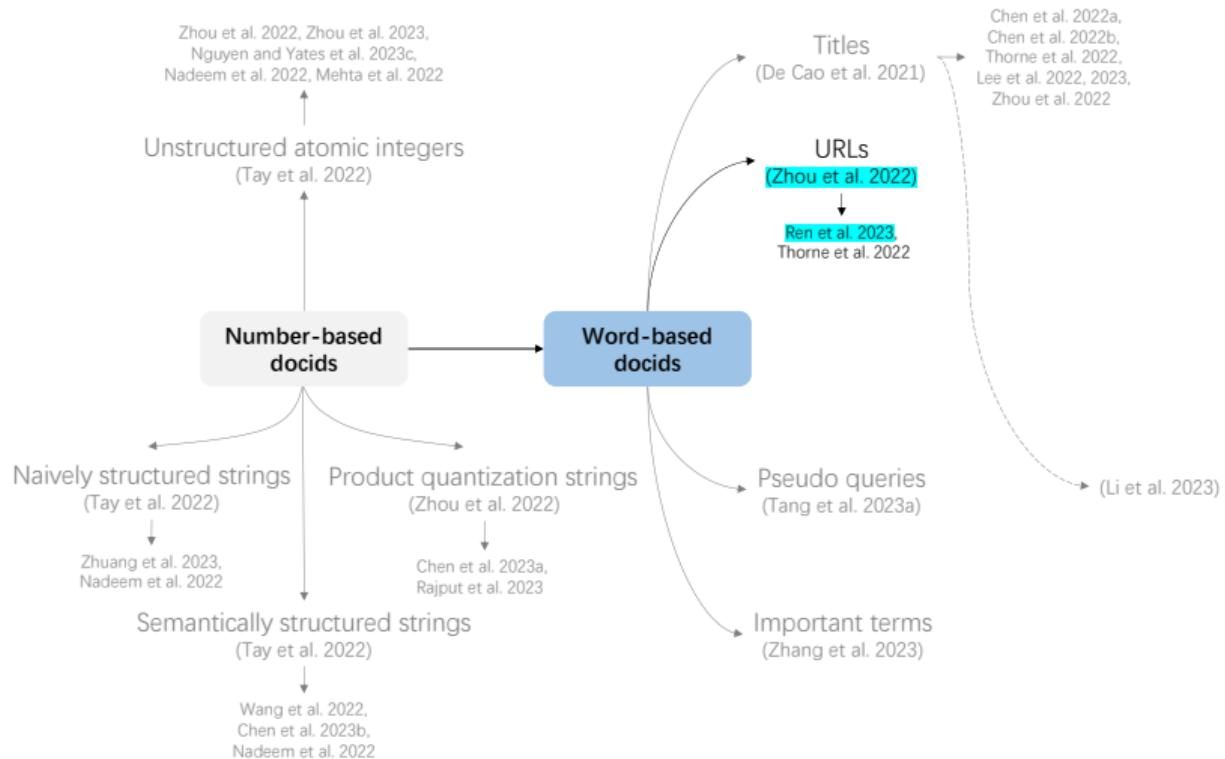
## ClueWeb09-B

Clarke et al. 2010

## Gov2

Clarke et al. 2004

# URLs and subsequent work



If the special document metadata is not available

It is necessary to design **automatic** docid generation techniques

## Word-based: Pseudo queries

- Doc2Query technique: pseudo queries are likely to be representative or related to the contents of documents

## Word-based: Pseudo queries

- Doc2Query technique: pseudo queries are likely to be representative or related to the contents of documents



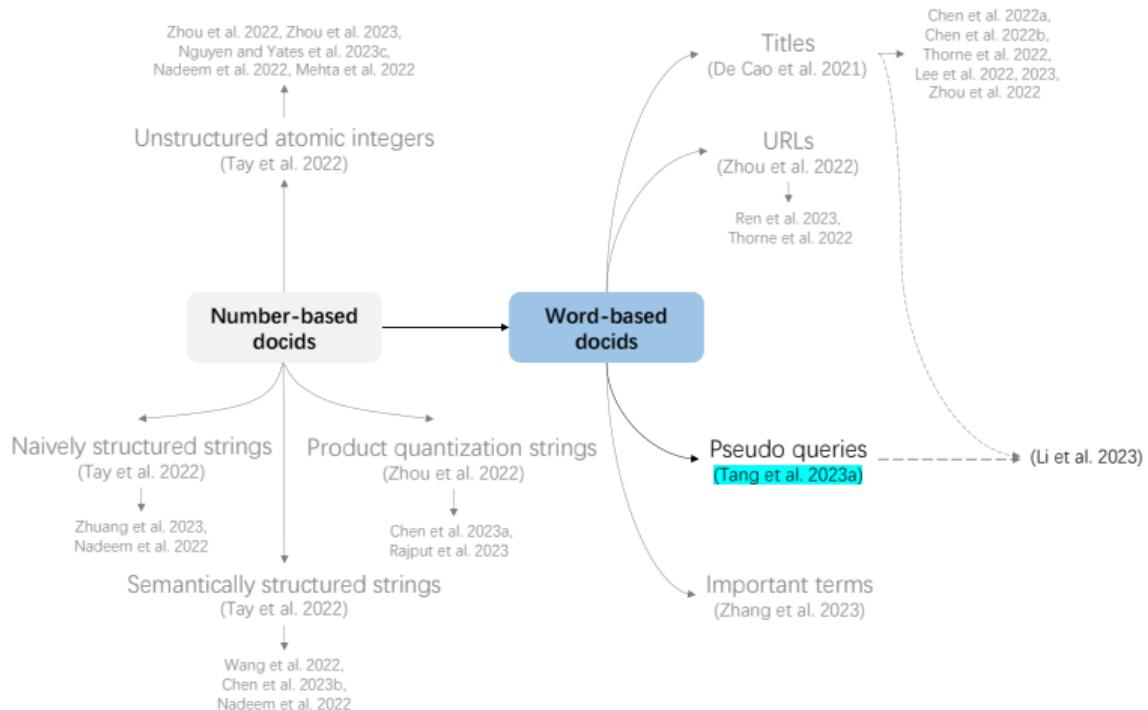
## Word-based: Pseudo queries

- Docid repetition problem
  - Tang et al. [2023a] uses the top 1 generated query as the docid for each document
  - Based on statistics, about 5% and 3% docids of documents are not unique in MS MARCO and Natural questions datasets, respectively
  - It is reasonable that different documents may share the same docid if they share very similar essential information

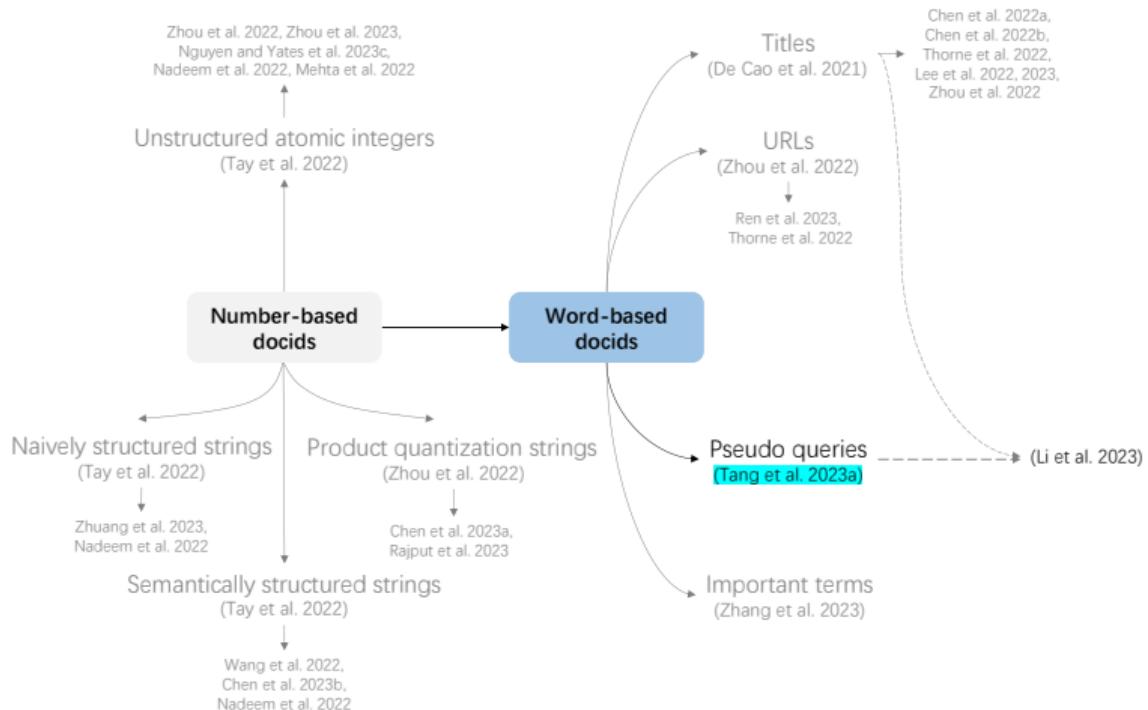
## Word-based: Pseudo queries

- Docid repetition problem
  - Tang et al. [2023a] uses the top 1 generated query as the docid for each document
  - Based on statistics, about 5% and 3% docids of documents are not unique in MS MARCO and Natural questions datasets, respectively
  - It is reasonable that different documents may share the same docid if they share very similar essential information
- Countermeasure
  - If a docid corresponds to multiple documents, return all of them in a random order, while keeping the relative order of documents corresponding to other docids

# Pseudo queries and subsequent work



# Pseudo queries and subsequent work



Without the requirements of certain document metadata, e.g., titles and URLs

## False pruning [Zhang et al., 2023]

Titles, URLs and pseudo queries:

## False pruning [Zhang et al., 2023]

Titles, URLs and pseudo queries:

- One pre-defined sequence

## False pruning [Zhang et al., 2023]

Titles, URLs and pseudo queries:

- One pre-defined sequence
- The requirement for the exact generation

## False pruning [Zhang et al., 2023]

Titles, URLs and pseudo queries:

- One pre-defined sequence
- The requirement for the exact generation
- The targeted document will be missed from the retrieval result if a false prediction about its identifier is made in any step of the generation process

**The permutation of docids becomes critical**

## Word-based: Important terms [Zhang et al., 2023]

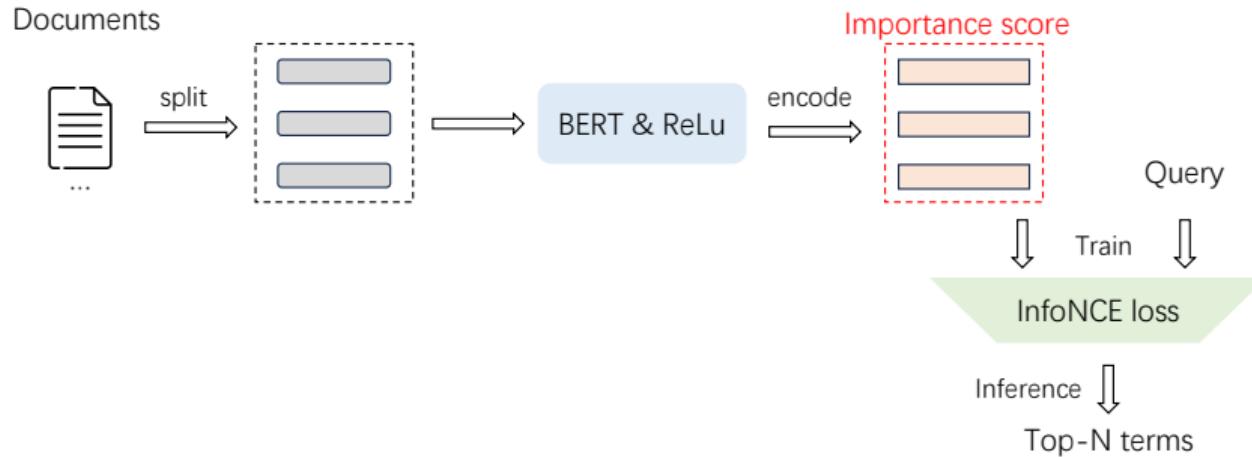
- Any permutation of the term set will be a valid identification for the corresponding document

## Word-based: Important terms [Zhang et al., 2023]

- Any permutation of the term set will be a valid identification for the corresponding document
- Important terms: A set of document terms that have high importance scores

## Important terms: AutoTSG [Zhang et al., 2023]

- Importance scores: The relevance scores of terms with respect to the query



## Important terms: AutoTSG [Zhang et al., 2023]

Docid repetition problem

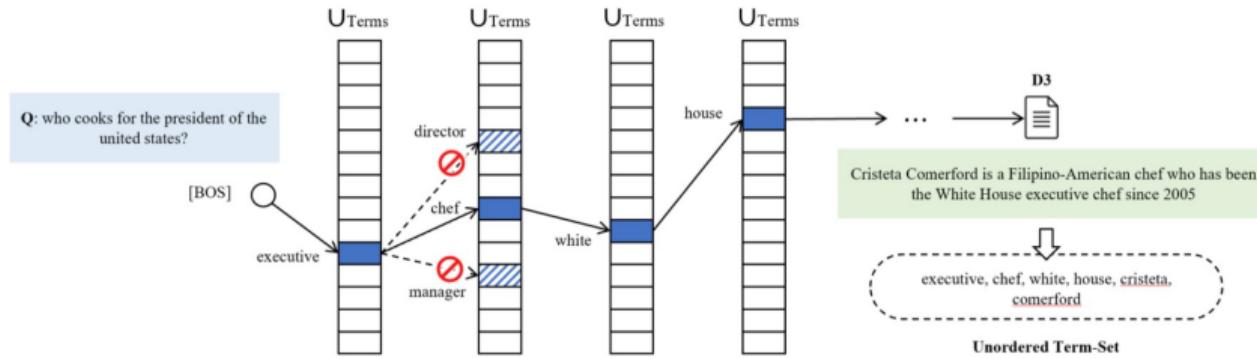
- If the number of terms is sufficiently large, all documents within the corpus can be unique

## Important terms: AutoTSG [Zhang et al., 2023]

Docid repetition problem

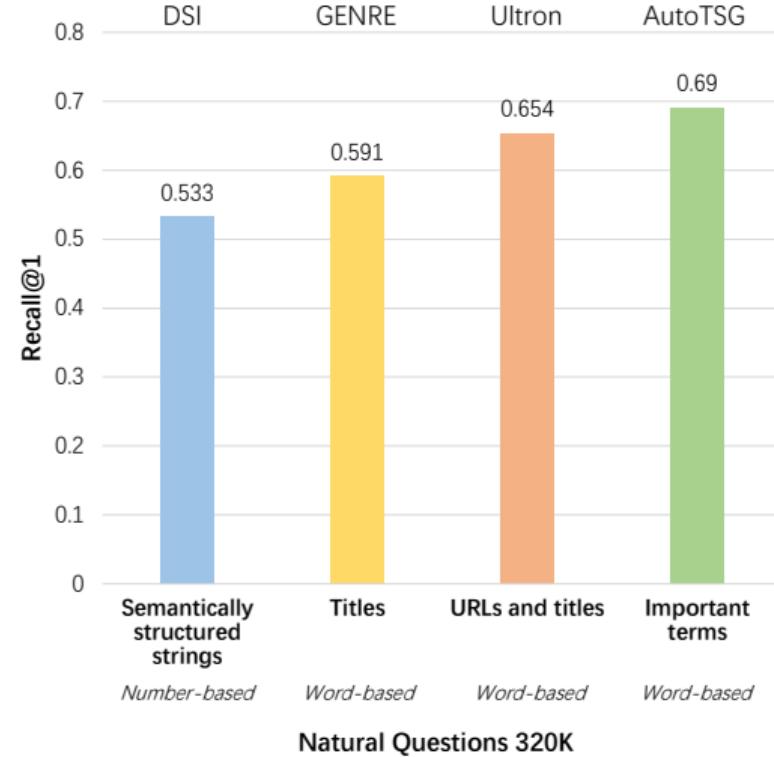
- If the number of terms is sufficiently large, all documents within the corpus can be unique
- For a moderate-scale corpus like Natural Questions, specifying 12 terms is already sufficient to ensure uniqueness

# Important terms: AutoTSG [Zhang et al., 2023]



- Any permutation of the term-set identifier will lead to the retrieval of the corresponding document

# Performance comparisons



- Backbone: T5-base
- Using important term sets obtained through relevance matching as docids help represent the important information of the document
- This method also mitigates the issue of false pruning

## Word-based docids: Summary



Semantically related to the content of the document

## Word-based docids: Summary



Semantically related to the content of the document



Good interpretability

## Word-based docids: Summary



Semantically related to the content of the document



Good interpretability



Rely on metadata or labeled data

## Word-based docids: Summary



Semantically related to the content of the document



Good interpretability

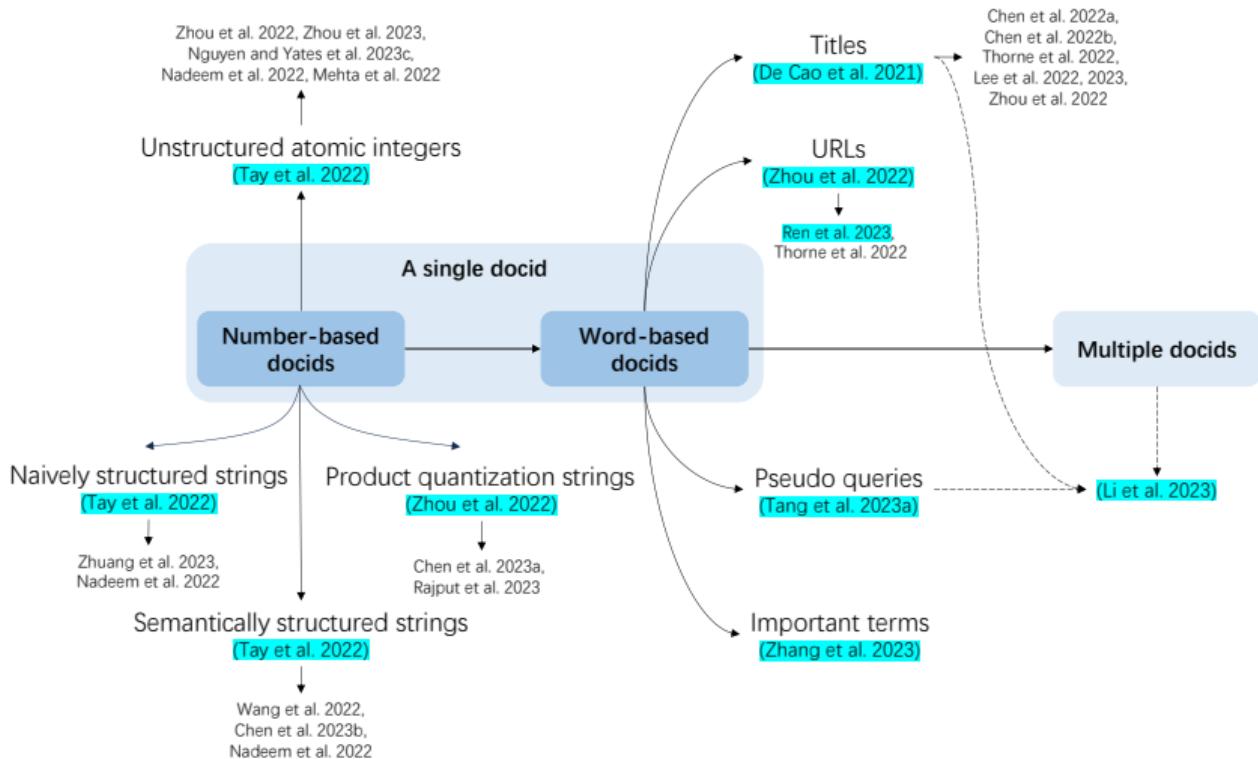


Rely on metadata or labeled data



May lead to duplication

# A single docid: Summary



## A single docid: Summary



The design of a single docid is relatively straightforward

## A single docid: Summary



The design of a single docid is relatively straightforward



The GR model may easily learn the one-to-one mapping relationship

## A single docid: Summary



The design of a single docid is relatively straightforward



The GR model may easily learn the one-to-one mapping relationship



Single identifiers are typically short strings, providing limited information about the document

## A single docid: Summary

-  The design of a single docid is relatively straightforward
-  The GR model may easily learn the one-to-one mapping relationship
-  Single identifiers are typically short strings, providing limited information about the document
-  A single type of identifier only represents a document from one view; and might be insufficient to effectively capture the entirety of the document's content

## Multiple docids

- Multiple identifiers can provide complementary information from different views

# Multiple docids

- Multiple identifiers can provide complementary information from different views

The screenshot shows the Wikipedia article for "Information retrieval". The page has a header "Information retrieval" and a sub-header "From Wikipedia, the free encyclopedia". Below the header, there are three main sections, each enclosed in a red box:

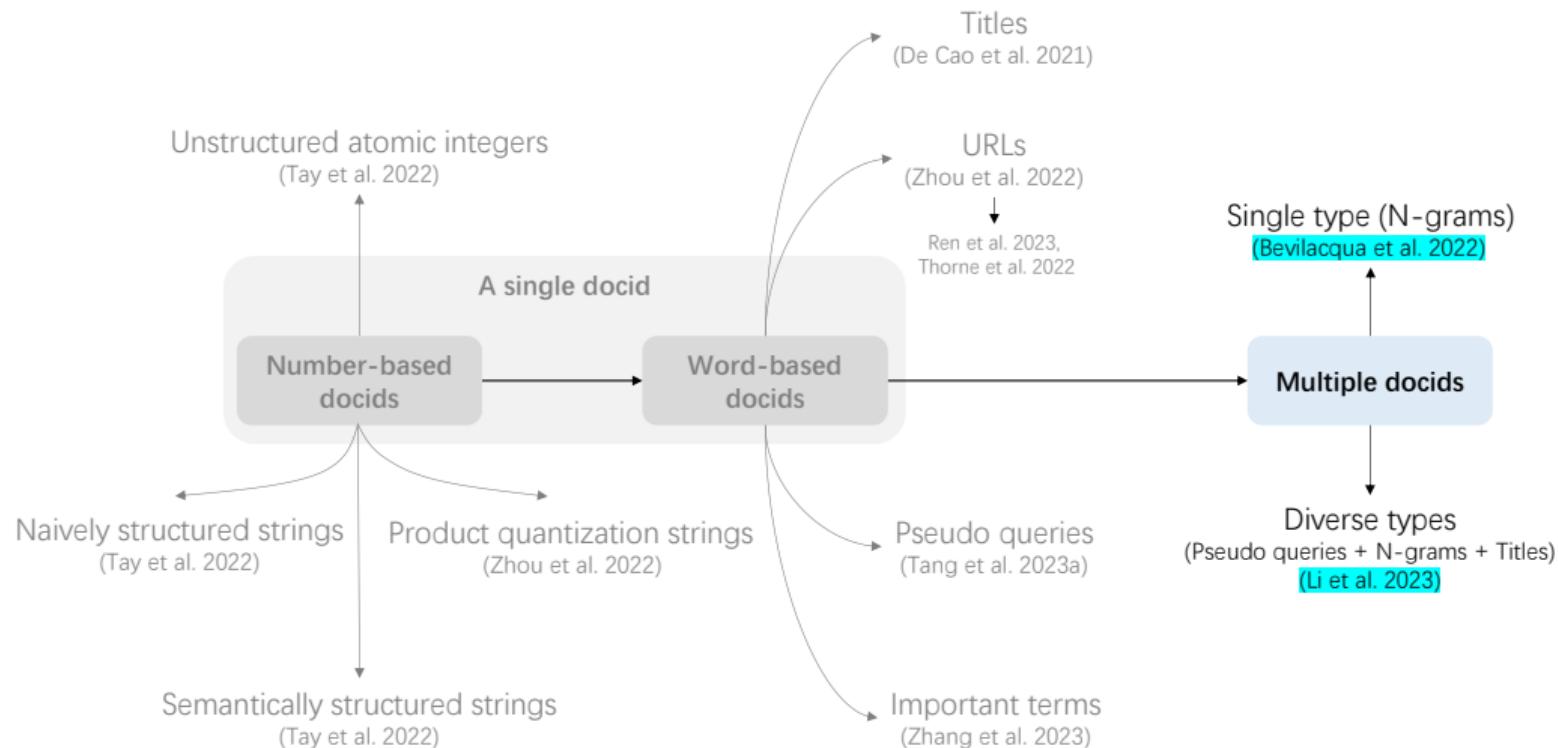
- Information retrieval**: A brief definition of information retrieval as the process of obtaining resources relevant to a user's query from a collection of documents.
- History**: A history of the development of information retrieval, mentioning Vannevar Bush, Emanuel Goldberg, and Gerard Salton.
- Applications**: A list of fields where information retrieval techniques are used, such as digital libraries and search engines.

Three arrows point from the right side of the slide to the right side of the highlighted sections:

- A dashed arrow points to the first section with the text "Overview of IR".
- A dashed arrow points to the second section with the text "History of IR".
- A dashed arrow points to the third section with the text "Applications of IR".

Multi-view information

# Multiple docids



## Multiple docids: Single type (N-grams) [Bevilacqua et al., 2022]

- All n-grams (i.e., substrings) in a document are treated as its possible identifiers

## Multiple docids: Single type (N-grams) [Bevilacqua et al., 2022]

- All n-grams (i.e., substrings) in a document are treated as its possible identifiers
- Part of n-grams as identifiers during training: Only the terms from the document that have **a high overlap with the query** are chosen as the target docids

### Carbon footprint

Carbon dioxide is released naturally by decomposition, ocean release and respiration. Humans contribute an **n-grams** increase of carbon dioxide emissions **by burning fossil fuels**, deforestation, and cement production. Methane ( $\text{CH}_4$ ) is largely released by coal, oil, and natural gas industries. Although methane is not mass-produced like carbon dioxide, it is still very prevalent.

## Multiple docids: Single type (N-grams) [Bevilacqua et al., 2022]

Docid repetition problem

- A **heuristic scoring function** is designed to address this **during inference**

## Multiple docids: Single type (N-grams) [Bevilacqua et al., 2022]

Docid repetition problem

- A **heuristic scoring function** is designed to address this **during inference**  
**We will discuss this in Section 5!**

## Multiple docids: Single type (Important n-grams) [Chen et al., 2023b]

- The **important n-grams** occurring in a document as its identifiers

## Multiple docids: Single type (Important n-grams) [Chen et al., 2023b]

- The **important n-grams** occurring in a document as its identifiers
- N-gram importance
  - Step 1: The query and its relevant document are concatenated with special delimiter tokens as a single input sequence
  - Step 2: Feed it into the original BERT model to get the [CLS] vector
  - Step 3: The token importance is computed by averaging the [CLS]-token attention weights
  - Step 4: The importance for the n-gram is the average of these tokens' importance

## Single type (Important n-grams) [Chen et al., 2023b]: An example

### ID for document retrieval Important n-grams

1. was an American entrepreneur,  
industrial designer
2. Jobs was forced out of Apple
3. He died of respiratory arrest  
related

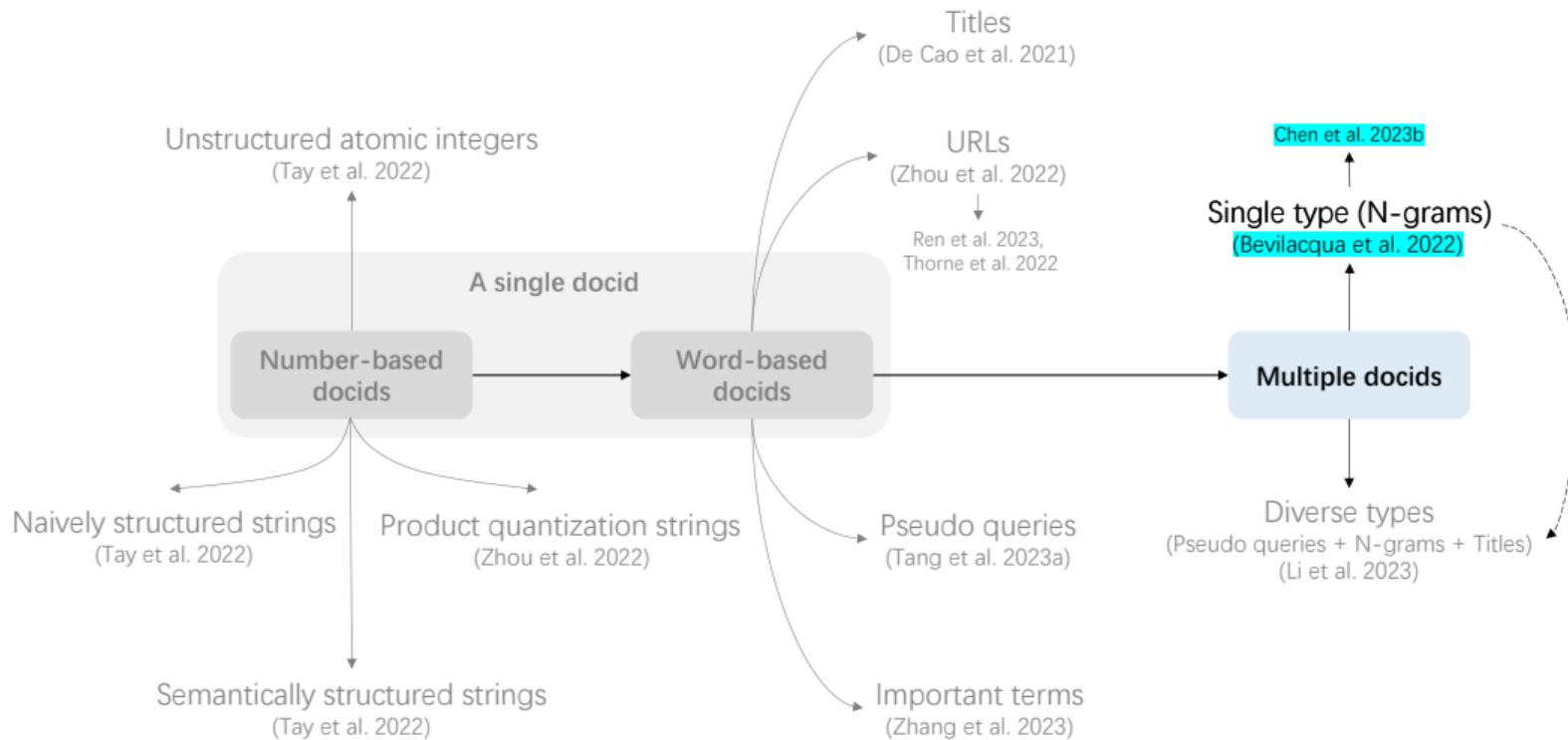
Steven Paul Jobs (February 24, 1955 – October 5, 2011)  
***was an American entrepreneur, industrial designer,***  
business magnate, media proprietor, and investor.

[...] In 1985, ***Jobs was forced out of Apple*** after a long power struggle with the company's board and its then-CEO John Sculley [...]

In 2003, Jobs was diagnosed with a pancreatic neuroendocrine tumor. ***He died of respiratory arrest related*** to the tumor on October 5, 2011 at the age of 56.

- Countermeasure for docid repetition problem: Similar to Bevilacqua et al. [2022]

# Single type (N-grams) and subsequent work



# Multiple docids: Diverse types (MINDER) [Li et al., 2023]

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

↑  
Relevant

**Passage** ([https://en.wikipedia.org/wiki/Does\\_He\\_Love\\_You](https://en.wikipedia.org/wiki/Does_He_Love_You))

"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. .....

## Multiview Identifiers

**Title:** Does He Love You

**Substrings:** "Does He Love You" is a song ..., **recorded as a duet by American country music artists** Reba McEntire and Linda Davis, ...

## Pseudo-queries:

Who wrote the song does he love you?

**Who sings does he love you?**

When was does he love you released by reba?

What is the first song in the album "Greatest Hits Volume Two" about?

- Three views of identifiers

# Multiple docids: Diverse types (MINDER) [Li et al., 2023]

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

↑  
Relevant

**Passage** ([https://en.wikipedia.org/wiki/Does\\_He\\_Love\\_You](https://en.wikipedia.org/wiki/Does_He_Love_You))

"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. .....

## Multiview Identifiers

**Title:** Does He Love You

**Substrings:** "Does He Love You" is a song ..., **recorded as a duet by American country music artists** Reba McEntire and Linda Davis, ...

## Pseudo-queries:

Who wrote the song does he love you?

**Who sings does he love you?**

When was does he love you released by reba?

What is the first song in the album "Greatest Hits Volume Two" about?

- Three views of identifiers
  - Title: Indicate the subject of a document

# Multiple docids: Diverse types (MINDER) [Li et al., 2023]

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

↑  
Relevant

**Passage** ([https://en.wikipedia.org/wiki/Does\\_He\\_Love\\_You](https://en.wikipedia.org/wiki/Does_He_Love_You))

"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. .....

## Multiview Identifiers

**Title:** Does He Love You

**Substrings:** "Does He Love You" is a song ..., **recorded as a duet by American country music artists** Reba McEntire and Linda Davis, ...

## Pseudo-queries:

Who wrote the song does he love you?

**Who sings does he love you?**

When was does he love you released by reba?

What is the first song in the album "Greatest Hits Volume Two" about?

- Three views of identifiers

- Title: Indicate the subject of a document
- Substrings (N-grams): Be also semantically related

# Multiple docids: Diverse types (MINDER) [Li et al., 2023]

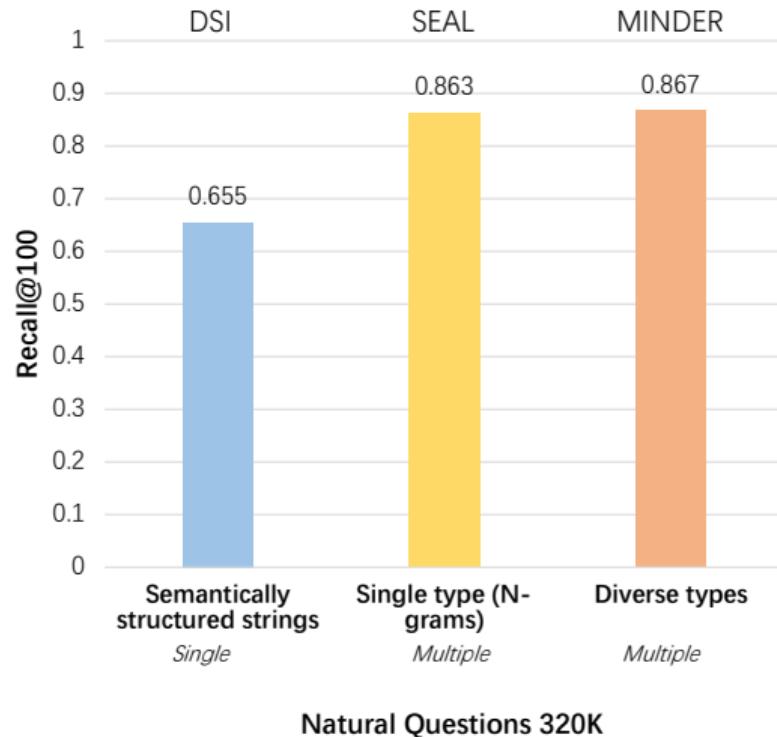
**Query:** Who is the singer of *does he love you?*

↑  
Relevant

Passage ( <a href="https://en.wikipedia.org/wiki/Does_He_Love_You">https://en.wikipedia.org/wiki/Does_He_Love_You</a> )
"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. ....
<b>Multiview Identifiers</b>
<b>Title:</b> Does He Love You
<b>Substrings:</b> "Does He Love You" is a song ..., <b>recorded as a duet by American country music artists</b> Reba McEntire and Linda Davis, ...
<b>Pseudo-queries:</b>
Who wrote the song does he love you?
<b>Who sings does he love you?</b>
When was does he love you released by reba?
What is the first song in the album "Greatest Hits Volume Two" about?

- Three views of identifiers
  - Title: Indicate the subject of a document
  - Substrings (N-grams): Be also semantically related
  - Pseudo-queries: Integrate multiple segments and contextualized information

## Performance comparisons



- Backbone: BART-large
- Results: Using multiple docids for a document yields better results than using a single docid

## Multiple docids: Summary



Multiple docids can provide a more **comprehensive** representation of the document, assisting the model in gaining a multifaceted understanding

## Multiple docids: Summary

- thumb up icon Multiple docids can provide a more **comprehensive** representation of the document, assisting the model in gaining a multifaceted understanding
- thumb up icon Similar docids across different documents can reflect the **similarity** between the documents

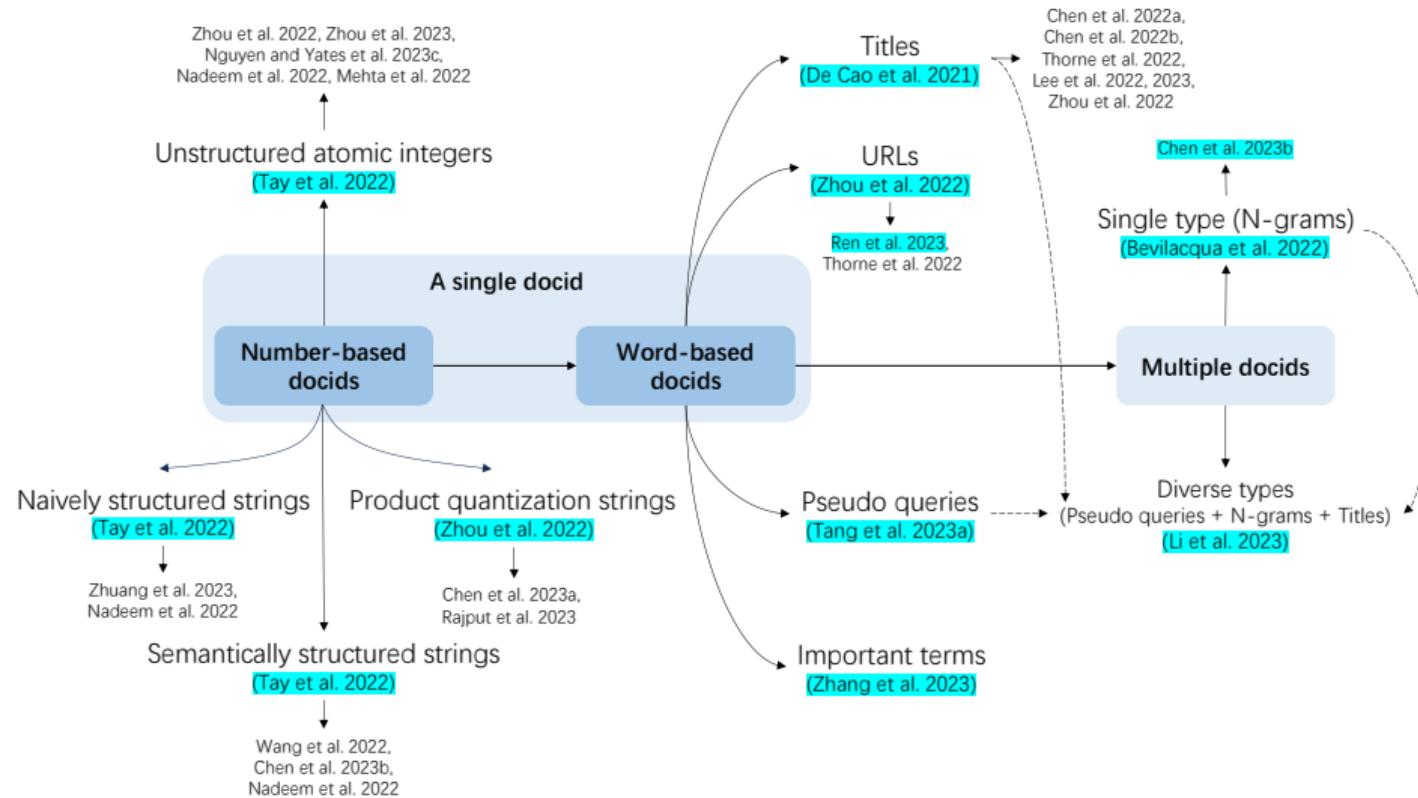
## Multiple docids: Summary

-  Multiple docids can provide a more **comprehensive** representation of the document, assisting the model in gaining a multifaceted understanding
-  Similar docids across different documents can reflect the **similarity** between the documents
-  GR models with the increased identifier numbers demand **more memory usage** and **inference time** compared to GR models with single identifiers

## Multiple docids: Summary

-  Multiple docids can provide a more **comprehensive** representation of the document, assisting the model in gaining a multifaceted understanding
-  Similar docids across different documents can reflect the **similarity** between the documents
-  GR models with the increased identifier numbers demand **more memory usage** and **inference time** compared to GR models with single identifiers
-  It is challenging to design **discriminative** multiple identifiers for a document

# Pre-defined static docids: Summary



# Pre-defined static docids: Summary

Docid type		Construction	Uniqueness	The degree of semantic connection to the document	Relying on labeled data	Relying on metadata
A single docid: Number-based	Unstructured atomic integers (Tay et al. 2022)	Easy	Yes	None	No	No
	Naively structured strings (Tay et al. 2022)	Easy	Yes	None	No	No
	Semantically structured strings (Tay et al. 2022)	Moderate	Yes	Weak	No	No
	Product quantization strings (Zhou et al. 2022)	Moderate	No	Moderate	No	No
A single docid: Word-based	Titles (De Cao et al. 2021)	Easy	No	Strong	No	Yes
	URLs (Zhou et al. 2022, Ren et al. 2023)	Easy	Yes	Strong	No	Yes
	Pseudo queries (Tang et al. 2023a)	Moderate	No	Strong	Yes	No
	Important terms (Zhang et al. 2023)	Hard	Yes	Strong	Yes	No
Multiple docids	Single type: N-grams (Bevilacqua et al. 2022)	Easy	No	Moderate	No	No
	Diverse types (Li et al. 2023)	Moderate	No	Strong	Yes	Yes

## Pre-defined static docids: Obvious constrains

They are fixed and not learnable by training on the retrieval tasks

## Pre-defined static docids: Obvious constrains

They are fixed and not learnable by training on the retrieval tasks



Not specifically optimized for retrieval tasks

## Pre-defined static docids: Obvious constrains

They are fixed and not learnable by training on the retrieval tasks



Not specifically optimized for retrieval tasks



Difficult to learn semantics and relationships between documents

**How to design learnable docids tailored for retrieval tasks?**

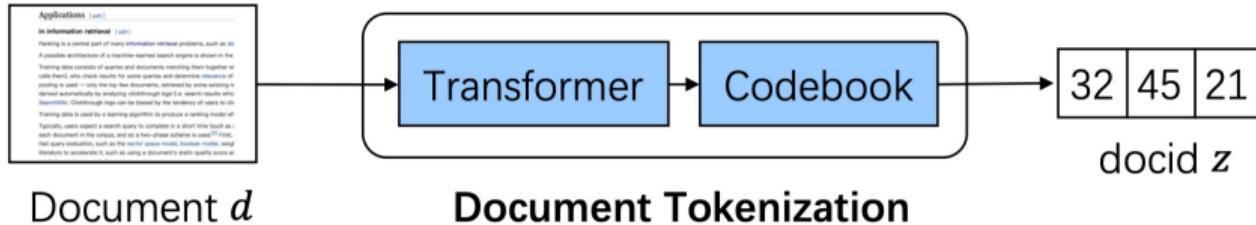
## Learnable docids

- GenRet [Sun et al., 2023] learns to tokenize documents into short discrete representations via a discrete auto-encoding, jointly training with the retrieval task

## Learnable docids

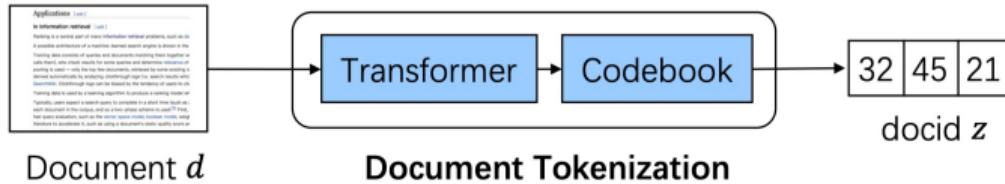
- GenRet [Sun et al., 2023] learns to tokenize documents into short discrete representations via a discrete auto-encoding, jointly training with the retrieval task
- NOVO [Wang et al., 2023] uses unique n-gram sets identifying each document and can be generated in any order and can be optimized through retrieval tasks

# Learnable docids: GenRet [Sun et al., 2023]



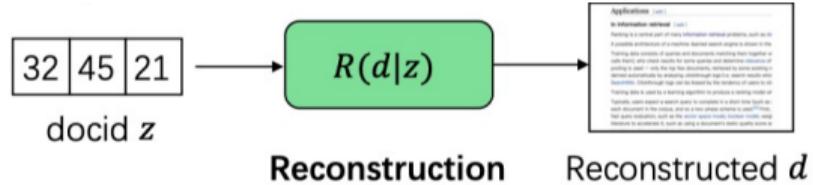
- Docid: A **sequence of discrete numbers** is the docid for a given document converted by a document tokenization model
- Training: **Jointly training** with a **document tokenization task**, **reconstruction task** and **retrieval task**

## Learnable docids: GenRet [Sun et al., 2023]



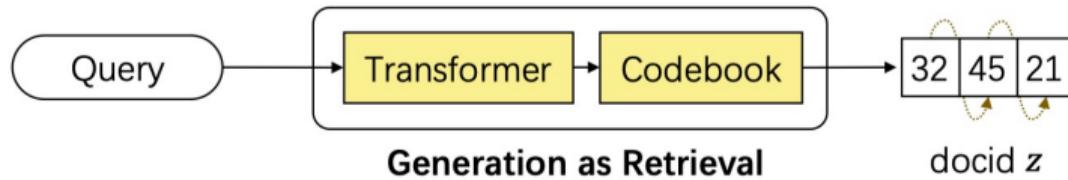
- Document tokenization task: Produce docids for documents

## Learnable docids: GenRet [Sun et al., 2023]



- Reconstruction task: Learn to reconstruct a document based on a docid

## Learnable docids: GenRet [Sun et al., 2023]



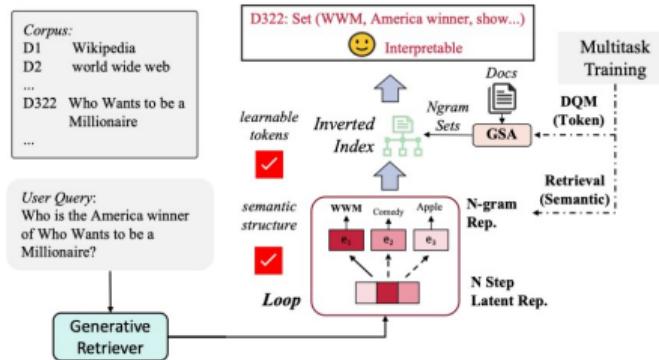
- Retrieval task: Generate relevant docids directly for a query

## Learnable docids: GenRet [Sun et al., 2023]

Docid repetition problem

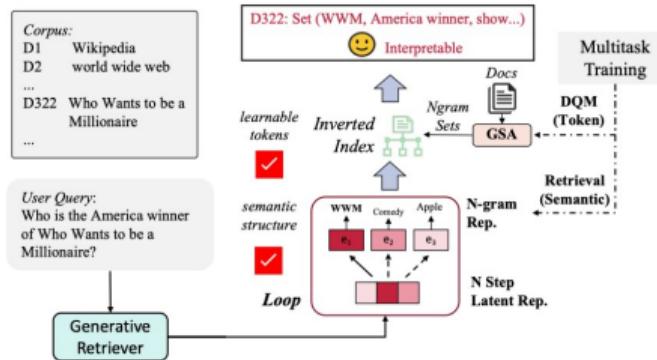
- All corresponding documents are retrieved and shuffled in **an arbitrary order**

# Learnable docids: NOVO [Wang et al., 2023]



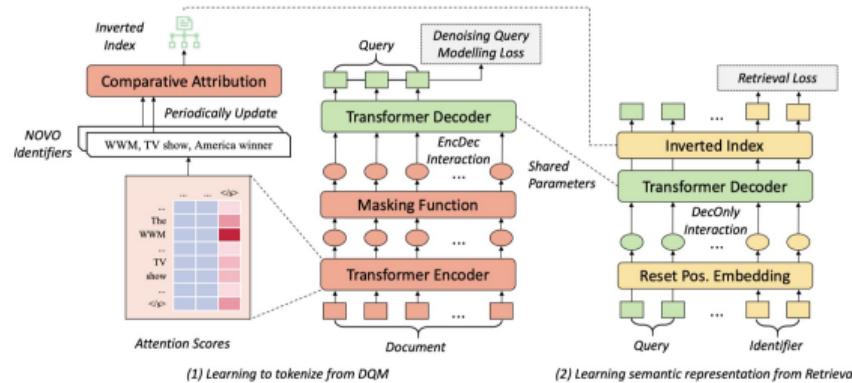
- Docid: **Unique n-grams sets** of the documents obtained from global self-attention

# Learnable docids: NOVO [Wang et al., 2023]



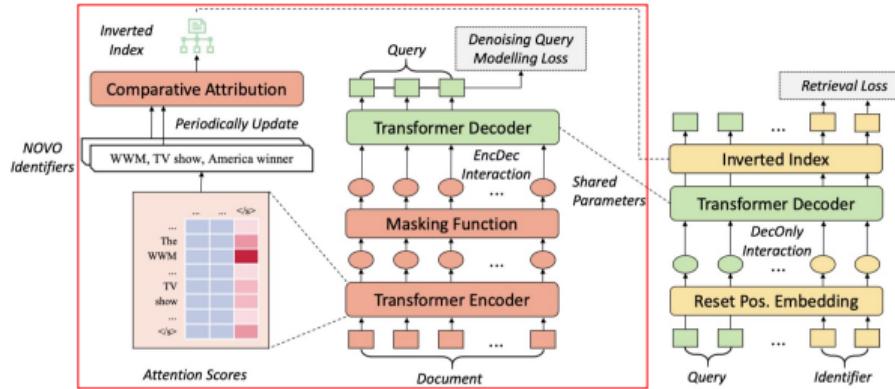
- Docid: **Unique n-grams sets** of the documents obtained from global self-attention
- Decoding: A document can be retrieved by generating its n-grams in the sets in any order

# Learnable docids: NOVO [Wang et al., 2023]



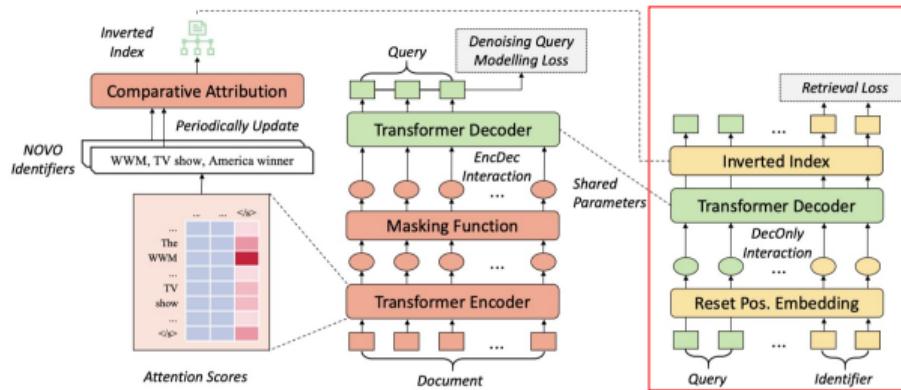
- Docids are learned by the denoising query modeling task and retrieval task jointly

# Learnable docids: NOVO [Wang et al., 2023]



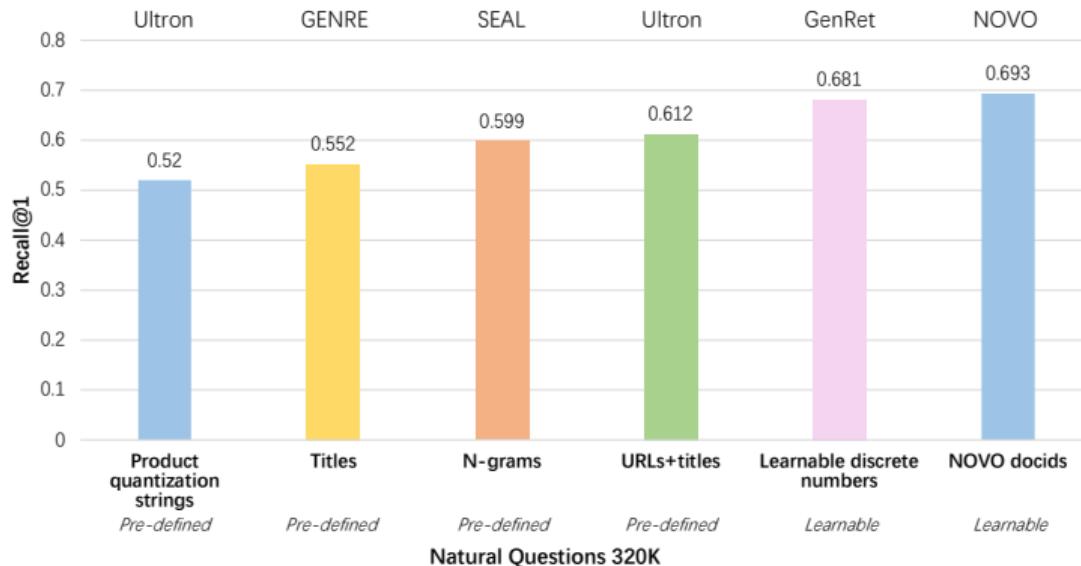
- **Denoising query modeling task:** By learning to generate queries with noisy documents, n-grams that are more relevant to the query are may be filtered out

# Learnable docids: NOVO [Wang et al., 2023]



- **Retrieval task:** The model learns the mapping from the query to relevant docids to update docid semantics

# Performance comparisons



- Backbone: T5-base
- Results: Two learnable docids yields better results than partial pre-defined static docids

## Learnable docids: Summary



It can be optimized together with the ultimate goal of GR to better adapt to retrieval

## Learnable docids: Summary

- thumb up icon It can be optimized together with the ultimate goal of GR to better adapt to retrieval
- thumb up icon A learnable approach can enable number-based docids like those in GenRet [Sun et al., 2023] to perform well

## Learnable docids: Summary

-  It can be optimized together with the ultimate goal of GR to better adapt to retrieval
-  A learnable approach can enable number-based docids like those in GenRet [[Sun et al., 2023](#)] to perform well
-  It relies on complex task design for learning

## Learnable docids: Summary

-  It can be optimized together with the ultimate goal of GR to better adapt to retrieval
-  A learnable approach can enable number-based docids like those in GenRet [[Sun et al., 2023](#)] to perform well
-  It relies on complex task design for learning
-  The learning process is complex, as docids change and require iterative learning

## Docid design: Summary

- Shall we use randomize numbers as the docids?
  - Random number strings can serve as docids, but their effectiveness is limited

## Docid design: Summary

- Shall we use randomize numbers as the docids?
  - Random number strings can serve as docids, but their effectiveness is limited
- How to construct proper docids for the documents?
  - Designing predefined or learnable docids based on the semantics of the documents

## Docid design: Summary

- Shall we use randomize numbers as the docids?
  - Random number strings can serve as docids, but their effectiveness is limited
- How to construct proper docids for the documents?
  - Designing predefined or learnable docids based on the semantics of the documents
- Would the choices of different docids affect the model performance(effectiveness, capacity, etc.)?
  - The length and quantity of docids both impact the effectiveness of the model's performance
  - The influence on capacity is yet to be explored

## Docid design: Summary

Docid type				
Pre-defined	Single	Number-based	- Simplified construction	- Low interpretability - Moderate performance
		Word-based	- High interpretability - Good performance	- Single-perspective representation of documents
	Multiple		- Comprehensive document representations - Better performance	- Slightly more complex construction
Learnable		- Adapting to GR objectives - Best performance		- Complex learning process

## Docid design: Summary

Docid type				
Pre-defined	Single	Number-based	- Simplified construction	- Low interpretability - Moderate performance
		Word-based	- High interpretability - Good performance	- Single-perspective representation of documents
	Multiple	 Comprehensive document representations - Better performance		- Slightly more complex construction
Learnable		 Adapting to GR objectives - Best performance		- Complex learning process

## Docid design: Summary

Docid type				
Pre-defined	Single	Number-based	- Simplified construction	- Low interpretability - Moderate performance
		Word-based	- High interpretability - Good performance	- Single-perspective representation of documents
	Multiple	 Comprehensive document representations - Better performance		- Slightly more complex construction
Learnable		 Adapting to GR objectives - Best performance		- Complex learning process

Based on these docids

Model training → **Section 4!**

Model inference → **Section 5!**

**Coffee break**

## **Section 4: Training approaches**

## Revisit the definition of generative retrieval

GR usually exploits a Seq2Seq encoder-decoder architecture to generate a ranked list of docids for an input query, in an autoregressive fashion

## Standard training objective

The commonly used training objective for both indexing and retrieval is **maximizing likelihood estimation** (MLE):

$$\begin{aligned}\mathcal{L}_{Global}(Q, D, I_D, I_Q; \theta) &= \mathcal{L}_{Indexing}(D, I_D; \theta) + \mathcal{L}_{Retrieval}(Q, I_Q; \theta) \\ &= - \sum_{d \in D} \log P(id \mid d; \theta) - \sum_{q \in Q} \sum_{id^q \in I_Q} \log P(id^q \mid q; \theta)\end{aligned}$$

---

## Different learning scenarios based on the corpus

$$\begin{aligned}\mathcal{L}_{Global}(Q, \underline{\mathcal{D}}, I_D, I_Q; \theta) &= \mathcal{L}_{Indexing}(\underline{\mathcal{D}}, I_D; \theta) + \mathcal{L}_{Retrieval}(Q, I_Q; \theta) \\ &= - \sum_{d \in \underline{\mathcal{D}}} \log P(id \mid d; \theta) - \sum_{q \in Q} \sum_{id^q \in I_Q} \log P(id^q \mid q; \theta)\end{aligned}$$

## Different learning scenarios based on the corpus

$$\begin{aligned}\mathcal{L}_{Global}(Q, \underline{\mathcal{D}}, I_D, I_Q; \theta) &= \mathcal{L}_{Indexing}(\underline{\mathcal{D}}, I_D; \theta) + \mathcal{L}_{Retrieval}(Q, I_Q; \theta) \\ &= -\sum_{d \in \underline{\mathcal{D}}} \log P(id \mid d; \theta) - \sum_{q \in Q} \sum_{id^q \in I_Q} \log P(id^q \mid q; \theta)\end{aligned}$$

- **Stationary scenarios:** The document collection is fixed
- **Dynamic scenarios:** Information changes and new documents emerge incrementally over time

## Stationary scenarios

$$\begin{aligned}\mathcal{L}_{Global}(\underline{Q}, D, I_D, \underline{I_Q}; \theta) &= \mathcal{L}_{Indexing}(D, I_D; \theta) + \mathcal{L}_{Retrieval}(\underline{Q}, \underline{I_Q}; \theta) \\ &= - \sum_{d \in D} \log P(id \mid d; \theta) - \sum_{q \in Q} \sum_{id^q \in I_Q} \log P(\textcolor{red}{id^q} \mid q; \theta)\end{aligned}$$

According to the **availability of labeled data**, the training approaches in stationary scenarios can be generally classified into:

- **Supervised learning methods**
- **Pre-training methods**

## Supervised learning: Basic training method

- Learn the indexing task first, and then learn retrieval tasks
  - Step 1:  $\mathcal{L}_{Indexing}(D, I_D; \theta) = -\sum_{d \in D} \log P(id \mid d; \theta)$
  - Step 2:  $\mathcal{L}_{Retrieval}(Q, I_Q; \theta) = -\sum_{q \in Q} \sum_{id^q \in I_Q} \log P(id^q \mid q; \theta)$

## Supervised learning: Basic training method

- Learn the indexing task first, and then learn retrieval tasks
  - Step 1:  $\mathcal{L}_{Indexing}(D, I_D; \theta) = -\sum_{d \in D} \log P(id | d; \theta)$
  - Step 2:  $\mathcal{L}_{Retrieval}(Q, I_Q; \theta) = -\sum_{q \in Q} \sum_{id^q \in I_Q} \log P(id^q | q; \theta)$
- Learn indexing and retrieval tasks simultaneously in a **multitask** fashion

$$\begin{aligned}\mathcal{L}_{Global}(Q, D, I_D, I_Q; \theta) &= \mathcal{L}_{Indexing}(D, I_D; \theta) + \mathcal{L}_{Retrieval}(Q, I_Q; \theta) \\ &= -\sum_{d \in D} \log P(id | d; \theta) - \sum_{q \in Q} \sum_{id^q \in I_Q} \log P(id^q | q; \theta)\end{aligned}$$

## Supervised learning: Basic training method

- Learn the indexing task first, and then learn retrieval tasks
  - Step 1:  $\mathcal{L}_{Indexing}(D, I_D; \theta) = -\sum_{d \in D} \log P(id | d; \theta)$
  - Step 2:  $\mathcal{L}_{Retrieval}(Q, I_Q; \theta) = -\sum_{q \in Q} \sum_{id^q \in I_Q} \log P(id^q | q; \theta)$
- Learn indexing and retrieval tasks simultaneously in a **multitask** fashion



$$\begin{aligned}\mathcal{L}_{Global}(Q, D, I_D, I_Q; \theta) &= \mathcal{L}_{Indexing}(D, I_D; \theta) + \mathcal{L}_{Retrieval}(Q, I_Q; \theta) \\ &= -\sum_{d \in D} \log P(id | d; \theta) - \sum_{q \in Q} \sum_{id^q \in I_Q} \log P(id^q | q; \theta)\end{aligned}$$

## Supervised learning: Multitask learning via MLE

- For a single docid representing a document
  - Indexing: Learn the relationships between document-docid pairs
  - Retrieval: Pair the query and the docid of each relevant document, and learn the relationships between query-docid pairs

## Supervised learning: Multitask learning via MLE

- For a single docid representing a document
  - Indexing: Learn the relationships between document-docid pairs
  - Retrieval: Pair the query and the docid of each relevant document, and learn the relationships between query-docid pairs
- For multiple docids representing a document
  - Indexing: Pair the document and its each corresponding docid, and then learn the relationships between document-docid pairs
  - Retrieval: Pair the query and each docid of each relevant document, and learn the relationships between query-docid pairs

## Limitation (1): Single document granularity

$$\begin{aligned}\mathcal{L}_{Global}(Q, D, I_D, I_Q; \theta) &= \underline{\mathcal{L}_{Indexing}(D, I_D; \theta)} + \mathcal{L}_{Retrieval}(Q, I_Q; \theta) \\ &= - \sum_{d \in D} \log P(id \mid \textcolor{red}{d}; \theta) - \sum_{q \in Q} \sum_{id^q \in I_Q} \log P(id^q \mid q; \theta)\end{aligned}$$

## Limitation (1): Single document granularity

$$\begin{aligned}\mathcal{L}_{Global}(Q, D, I_D, I_Q; \theta) &= \underline{\mathcal{L}_{Indexing}(D, I_D; \theta)} + \mathcal{L}_{Retrieval}(Q, I_Q; \theta) \\ &= - \sum_{d \in D} \log P(id \mid \textcolor{red}{d}; \theta) - \sum_{q \in Q} \sum_{id^q \in I_Q} \log P(id^q \mid q; \theta)\end{aligned}$$

When indexing, memorizing each document at a single granularity, e.g., first  $L$  tokens or the full text, is **insufficient**, especially for long documents with rich semantics.

## Supervised learning: Multi-granularity enhanced

- Given a document, the **important passages  $p$**  and **sentences  $s$**  are selected to augment the indexing data

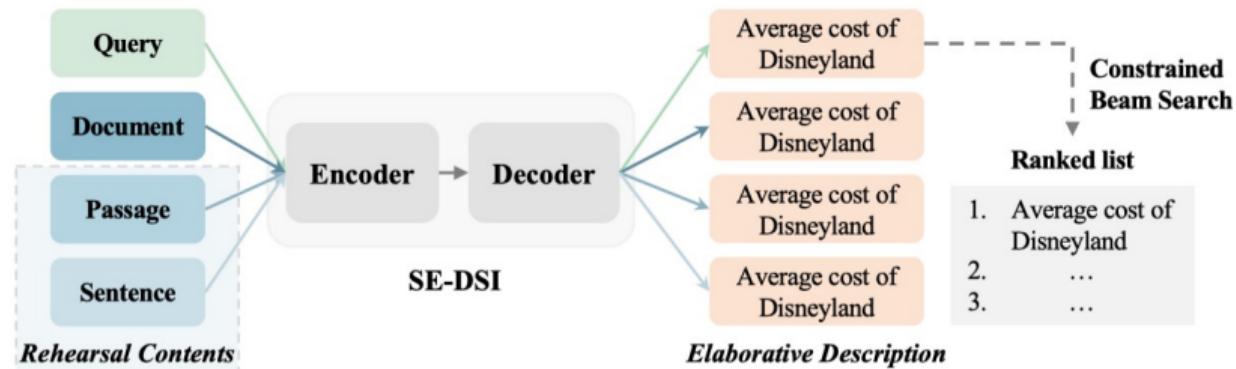
## Supervised learning: Multi-granularity enhanced

- Given a document, the **important passages  $p$**  and **sentences  $s$**  are selected to augment the indexing data

$$\mathcal{L}_{Indexing}(D, I_D; \theta) = -\left(\sum_{d \in D} \log P(id | \textcolor{red}{d}; \theta) + \sum_{p \in d} \log P(id | \textcolor{red}{p}; \theta) + \sum_{s \in d} \log P(id | \textcolor{red}{s}; \theta)\right)$$

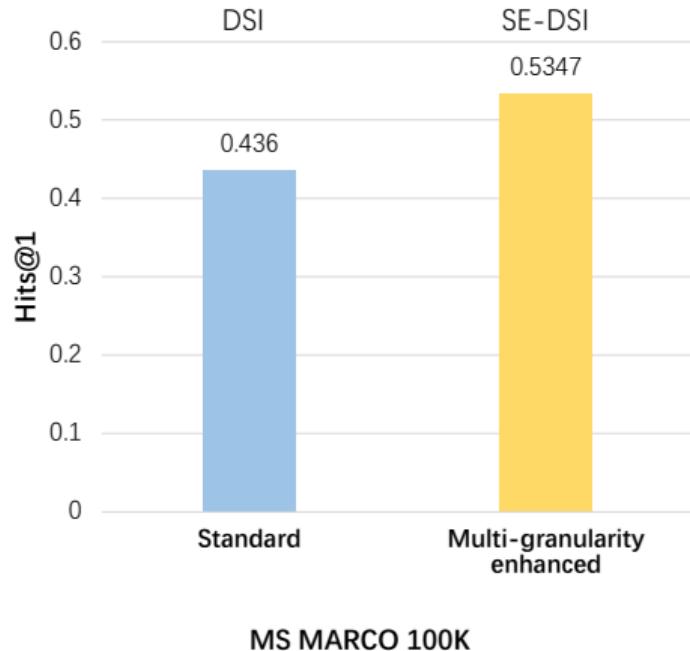
## Supervised learning: Multi-granularity enhanced

- Leading-style: Directly use the leading passages and sentences
- Summarization-style: Leverage the document summarization technique, e.g., TextRank, to highlight important parts



# Comparisons

Data source: Tang et al. [2023a]



- Backbone: T5-base
- Multi-granularity representations of documents can comprehensively encode the documents, and further contribute to the retrieval

## Limitation (2): The gap between indexing and retrieval

$$\begin{aligned}\mathcal{L}_{Global}(Q, D, I_D, I_Q; \theta) &= \mathcal{L}_{Indexing}(\underline{D}, I_D; \theta) + \mathcal{L}_{Retrieval}(\underline{Q}, I_Q; \theta) \\ &= - \sum_{d \in D} \log P(id \mid \underline{d}; \theta) - \sum_{q \in Q} \sum_{id^q \in I_Q} \log P(id^q \mid \underline{q}; \theta)\end{aligned}$$

## Limitation (2): The gap between indexing and retrieval

$$\begin{aligned}\mathcal{L}_{Global}(Q, D, I_D, I_Q; \theta) &= \mathcal{L}_{Indexing}(\underline{D}, I_D; \theta) + \mathcal{L}_{Retrieval}(\underline{Q}, I_Q; \theta) \\ &= - \sum_{d \in D} \log P(id \mid \underline{d}; \theta) - \sum_{q \in Q} \sum_{id^q \in I_Q} \log P(id^q \mid \underline{q}; \theta)\end{aligned}$$

Long document in indexing V.S. Short query in retrieval

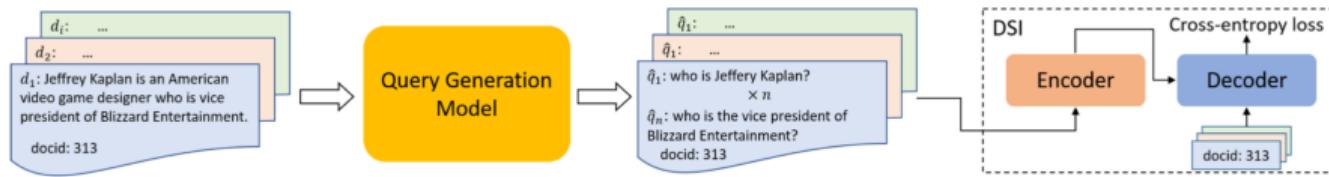
## Limitation (2): The gap between indexing and retrieval

$$\begin{aligned}\mathcal{L}_{Global}(Q, D, I_D, I_Q; \theta) &= \mathcal{L}_{Indexing}(\underline{D}, I_D; \theta) + \mathcal{L}_{Retrieval}(\underline{Q}, I_Q; \theta) \\ &= - \sum_{d \in D} \log P(id \mid \underline{d}; \theta) - \sum_{q \in Q} \sum_{id^q \in I_Q} \log P(id^q \mid \underline{q}; \theta)\end{aligned}$$

Long document in indexing V.S. Short query in retrieval

The data distribution mismatch that occurs between the indexing and retrieval

## Supervised learning: Pseudo query enhanced



Using a set of **pseudo queries  $pq$**  generated from the document as the inputs of the indexing task

## Supervised learning: Pseudo query enhanced

$$\mathcal{L}_{Indexing}(D, I_D; \theta) = - \sum_{d \in D} \log P(id | \underline{\textcolor{red}{d}}; \theta)$$


$$\mathcal{L}_{Indexing}(D, I_D; \theta) = - \sum_{pq \in D} \log P(id | \underline{\textcolor{red}{pq}}; \theta)$$

## Supervised learning: Pseudo query enhanced

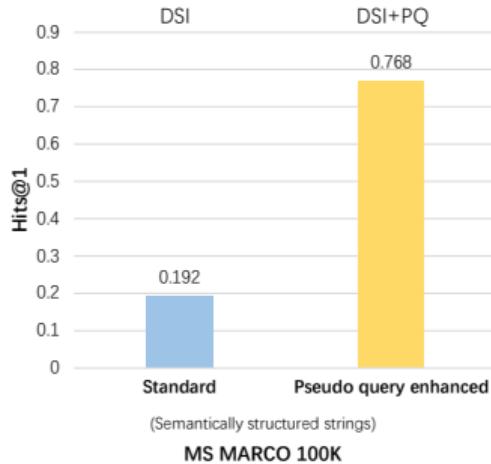
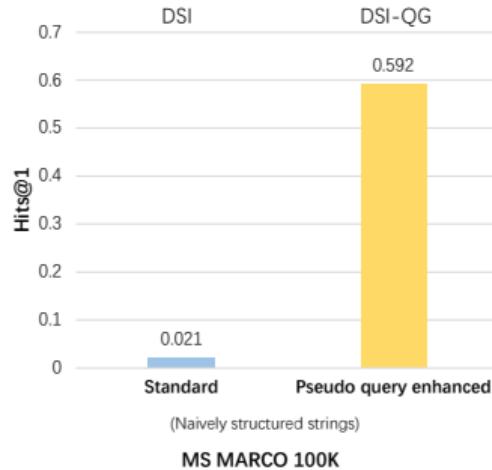
$$\mathcal{L}_{Indexing}(D, I_D; \theta) = - \sum_{d \in D} \log P(id | \underline{d}; \theta)$$


$$\mathcal{L}_{Indexing}(D, I_D; \theta) = - \sum_{pq \in D} \log P(id | \underline{pq}; \theta)$$

$$\mathcal{L}_{Retrieval}(Q, I_Q; \theta) = - \sum_{q \in Q} \sum_{id^q \in I_Q} \log P(id^q | \underline{q}; \theta)$$

# Comparisons

Data source: Pradeep et al. [2023], Zhuang et al. [2023]



- Backbone: T5-base
- Using only pseudo synthetic queries to docid during indexing is an effective training strategy on MS MARCO [Pradeep et al., 2023]

## Limitation (3): Limited labeled data

$$\mathcal{L}_{Global}(Q, D, I_D, I_Q; \theta) = \mathcal{L}_{Indexing}(D, I_D; \theta) + \boxed{\mathcal{L}_{Retrieval}(Q, I_Q; \theta)} ?$$

## Limitation (3): Limited labeled data

$$\mathcal{L}_{Global}(Q, D, I_D, I_Q; \theta) = \mathcal{L}_{Indexing}(D, I_D; \theta) + \boxed{\mathcal{L}_{Retrieval}(Q, I_Q; \theta)} ?$$

What should we do if there is no or few labeled query-docid pairs?

## Pre-training methods

Constructing **pseudo query-docid pairs**  $(PQ, I_Q^P)$  for the **pre-training** retrieval task

$$\mathcal{L}_{Pre-train}(PQ, D, I_D, I_Q^P; \theta) = \mathcal{L}_{Indexing}(D, I_D; \theta) + \underline{\mathcal{L}_{Retrieval}(PQ, I_Q^P; \theta)}$$

# CorpusBrain [Chen et al., 2022b]: Pre-training

INPUT:  
Apple Inc. is an American multinational  
[...] software and online services.

OUTPUT:  
Apple Inc. ISS

INPUT:  
Apple was founded as Apple Computer  
Company on April 1, 1976 [...] while  
Jobs resigned to found NeXT, taking  
some Apple employees with him.

OUTPUT:  
Apple Inc. [SEP] Tim Cook LPS

## Apple Inc.

Apple Inc. is an American multinational technology company that specializes in consumer electronics, software and online services. Apple is the largest information technology company by revenue [...]

Apple was founded as Apple Computer Company on April 1, 1976, by Steve Jobs, Steve Wozniak and Ronald Wayne to develop and sell Wozniak's Apple I personal computer [...] Apple went public in 1980, to instant financial success. The company developed computers featuring innovative graphical user interfaces, including the original Macintosh, announced in a critically acclaimed advertisement, "1984", directed by Ridley Scott. By 1985, the high cost of its products and power struggles between executives caused problems. Wozniak stepped back from Apple amicably, while Jobs resigned to found NeXT, taking some Apple employees with him.

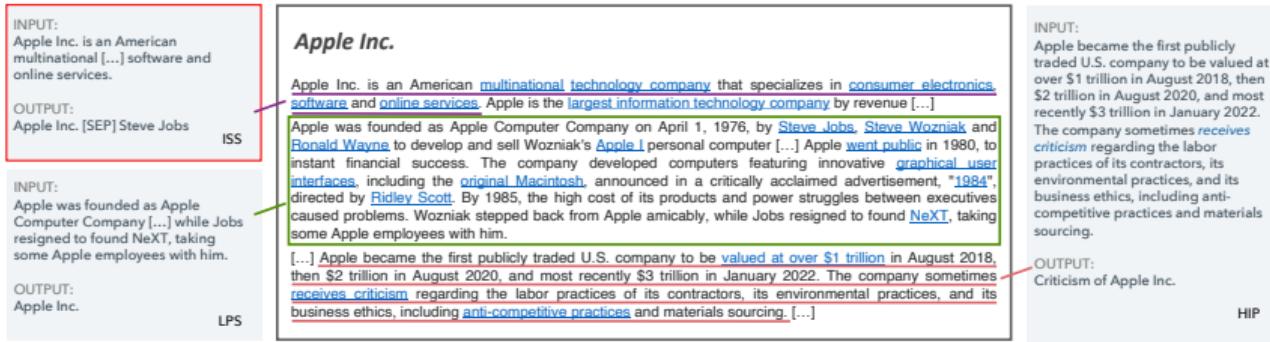
[...] Apple became the first publicly traded U.S. company to be valued at over \$1 trillion in August 2018, then \$2 trillion in August 2020, and most recently \$3 trillion in January 2022. The company sometimes receives criticism regarding the labor practices of its contractors, its environmental practices, and its business ethics, including anti-competitive practices and materials sourcing. [...]

INPUT:  
Apple became the first publicly traded U.S. company to be valued at over \$1 trillion in August 2018, then \$2 trillion in August 2020, and most recently \$3 trillion in January 2022. The company sometimes receives criticism regarding the labor practices of its contractors, its environmental practices, and its business ethics, including anti-competitive practices and materials sourcing.

OUTPUT:  
Criticism of Apple Inc. HIP

Based on Wikipedia, three pre-training retrieval tasks are constructed

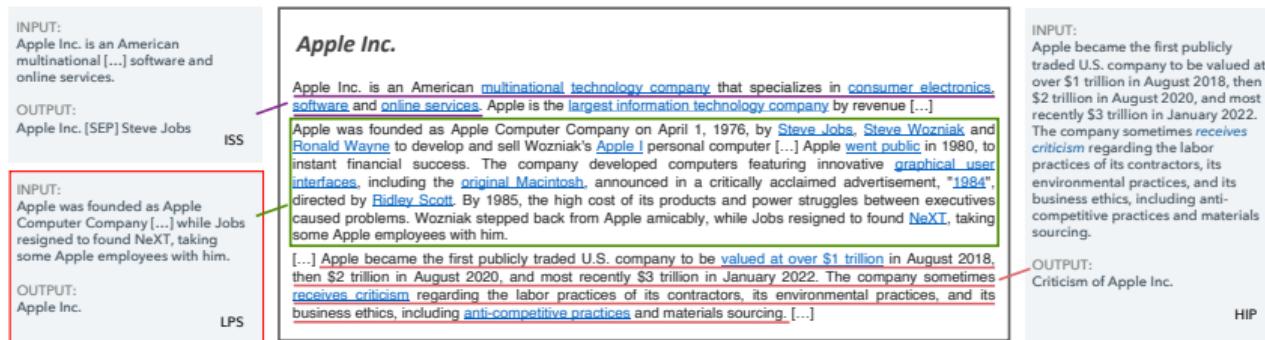
# CorpusBrain [Chen et al., 2022b]: Pre-training



## Inner Sentence Selection (ISS):

- Pseudo query ( $PQ$ ): Randomly selected **inner sentence** from its document
- Docid ( $I_Q^P$ ): Concatenated relevant **document titles**, i.e., “title [SEP] title [SEP] title”

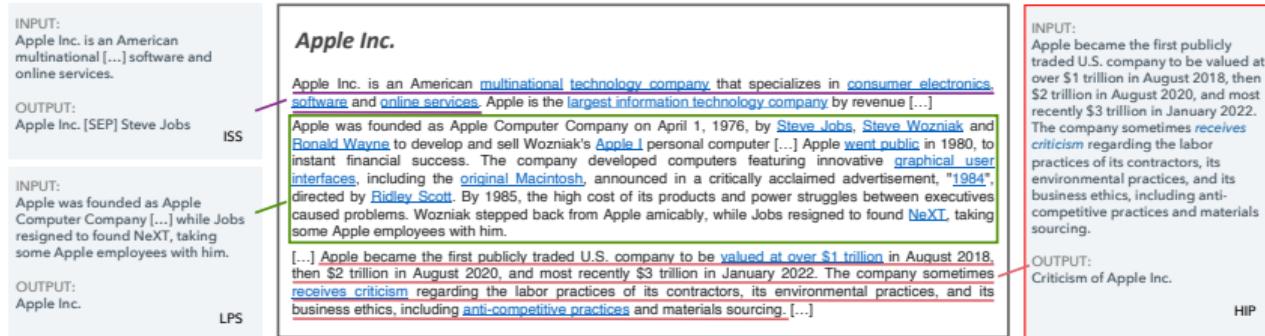
# CorpusBrain [Chen et al., 2022b]: Pre-training



## Lead Paragraph Selection (LPS):

- Pseudo query ( $PQ$ ): A (lead) **paragraph** is sampled from the document
- Docid ( $I_Q^P$ ): Concatenated relevant **document titles**

# CorpusBrain [Chen et al., 2022b]: Pre-training



## Hyperlink Identifier Prediction (HIP):

- Pseudo query ( $PQ$ ): The **anchor context**, i.e., the surrounding contextual information in the anchor's corresponding sentence
- Docid ( $I_Q^P$ ): The **document title** of the destination page

## CorpusBrain [Chen et al., 2022b]: Training and inference

- **Pre-training:** Based on the three pre-training tasks, a large number of pseudo pairs of query and document identifiers are constructed. All the tasks are formulated by a standard seq2seq objective for the pre-training

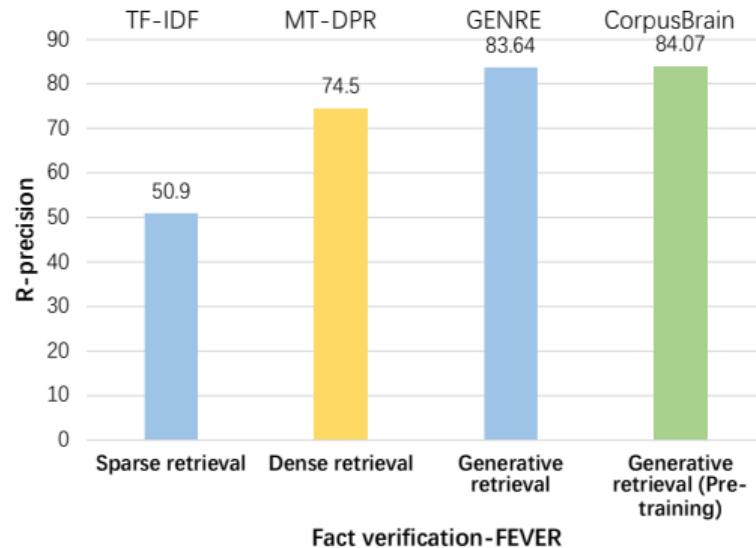
## CorpusBrain [Chen et al., 2022b]: Training and inference

- **Pre-training:** Based on the three pre-training tasks, a large number of pseudo pairs of query and document identifiers are constructed. All the tasks are formulated by a standard seq2seq objective for the pre-training
- **Fine-tuning:** CorpusBrain is fine-tuned using the processed data (in a Seq2Seq pair format) in downstream tasks

## CorpusBrain [Chen et al., 2022b]: Training and inference

- **Pre-training:** Based on the three pre-training tasks, a large number of pseudo pairs of query and document identifiers are constructed. All the tasks are formulated by a standard seq2seq objective for the pre-training
- **Fine-tuning:** CorpusBrain is fine-tuned using the processed data (in a Seq2Seq pair format) in downstream tasks
- **Test:** Given a test query, the fine-tuned CorpusBrain utilizes constrained beam search to decode relevant docids

## CorpusBrain [Chen et al., 2022b]: Performance



- In the KILT leaderboard, Corpusbrain achieved first place in 5 of them, second place in 1 task, and third place in 4 tasks, outperforming traditional pipelined approaches

## Challenge (4): Pointwise optimization for GR

$$\begin{aligned}\mathcal{L}_{Global}(Q, D, I_D, I_Q; \theta) &= \mathcal{L}_{Indexing}(D, I_D; \theta) + \underline{\mathcal{L}_{Retrieval}(Q, I_Q; \theta)} \\ &= - \sum_{d \in D} \log P(id \mid d; \theta) - \sum_{q \in Q} \sum_{id^q \in I_Q} \log P(id^q \mid q; \theta)\end{aligned}$$

---

## Challenge (4): Pointwise optimization for GR

$$\begin{aligned}\mathcal{L}_{Global}(Q, D, I_D, I_Q; \theta) &= \mathcal{L}_{Indexing}(D, I_D; \theta) + \underline{\mathcal{L}_{Retrieval}(Q, I_Q; \theta)} \\ &= -\sum_{d \in D} \log P(id \mid d; \theta) - \sum_{q \in Q} \sum_{id^q \in I_Q} \log P(id^q \mid q; \theta)\end{aligned}$$

---

- It assumes the likelihood for each relevant docid is **independent** of the other docids in the list for a query
- Ranking is a prediction task on **list of objects**

## Challenge (4): Pointwise optimization for GR

$$\begin{aligned}\mathcal{L}_{Global}(Q, D, I_D, I_Q; \theta) &= \mathcal{L}_{Indexing}(D, I_D; \theta) + \underline{\mathcal{L}_{Retrieval}(Q, I_Q; \theta)} \\ &= -\sum_{d \in D} \log P(id \mid d; \theta) - \sum_{q \in Q} \sum_{id^q \in I_Q} \log P(id^q \mid q; \theta)\end{aligned}$$

---

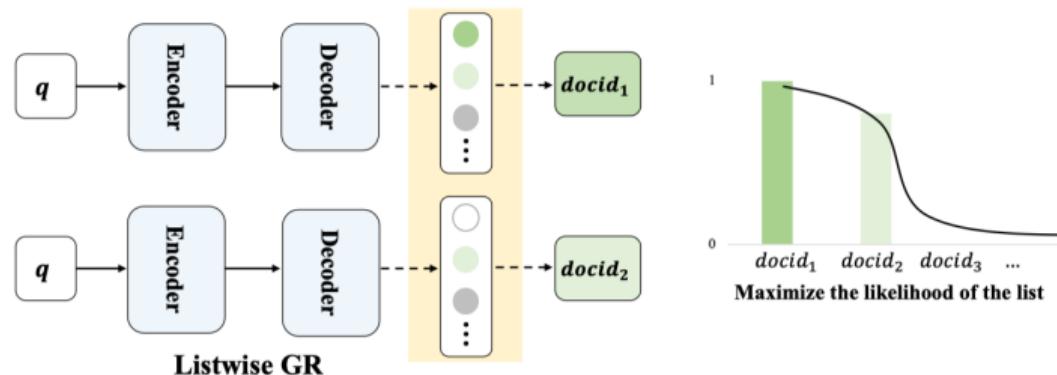
- It assumes the likelihood for each relevant docid is **independent** of the other docids in the list for a query
- Ranking is a prediction task on **list of objects**

Listwise optimization for GR is necessary!

# Listwise optimization for GR [Tang et al., 2023b]

Training with position-aware ListMLE

- View the docid ranking problem as a **sequential learning process**, with each step targeting to maximize the corresponding stepwise probability distribution



## Listwise optimization for GR [Tang et al., 2023b]

Given:

- A query  $q$
- Its ground-truth docid list  $\pi_q = [id^{(1)}, id^{(2)}, \dots]$ , in descending order of relevance, where  $id^{(1)}$  is the docid ranked at the first position, and  $id^{(2)}$  is the docid ranked at the second position, and so on

## Sequential learning process

**Step 1:** Maximize the following top-1 positional conditional probability:

$$P(id^{(1)} | q; \theta) = \frac{\exp(\tilde{P}(id^{(1)} | q; \theta))}{\sum_{j=1}^n \exp(\tilde{P}(id^{(j)} | q; \theta))},$$

where  $\tilde{P}(id^{(i)} | q; \theta) = \frac{\log P(id^{(i)} | q; \theta)}{|id^{(i)}|}$ , and  $P(id^{(i)} | q; \theta)$  is the generated likelihood of the  $i$ -th relevant docid  $id^{(i)}$  for  $q$

## Sequential learning process

**Step 2:** For  $i = 2, \dots, n$ , maximize the following  $i$ -th positional conditional probability given the preceding top  $i - 1$  docids,

$$P(id^{(i)} | q, id^{(1)}, \dots, id^{(i-1)}; \theta) = \frac{\exp(\tilde{P}(id^{(i)} | q; \theta))}{\sum_{j=i}^n \exp(\tilde{P}(id^{(j)} | q; \theta))}$$

The learning process ends at step  $n + 1$

## Listwise loss with position importance

- Listwise probability with position importance

$$\min_{\theta} - \log P(\pi_q | q; \theta)$$

$$= -\alpha(1) \log P(id^{(1)} | q; \theta) - \sum_{i=2}^n \alpha(i) \log P(id^{(i)} | q, id^{(1)}, \dots, id^{(i-1)}; \theta),$$

where the weight  $\alpha(\cdot)$  is a decreasing function

- Listwise loss function incorporating the probability based on Plackett-Luce model

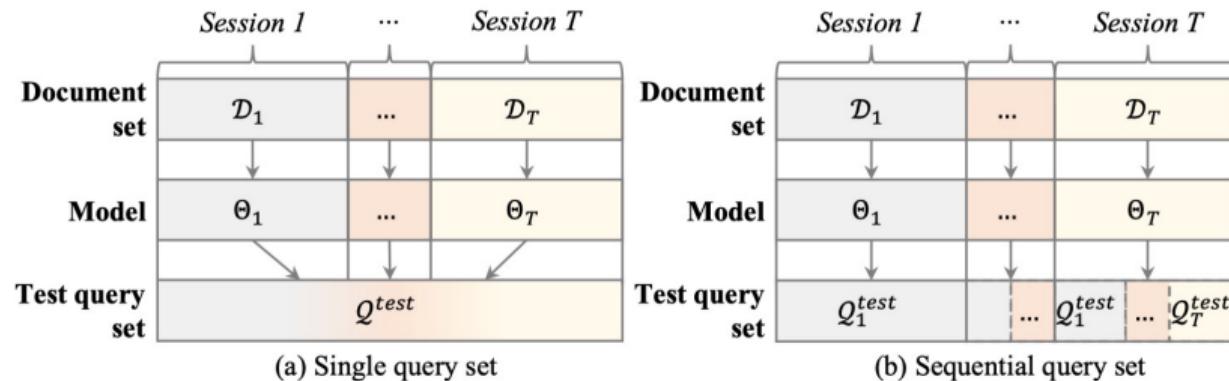
$$\mathcal{L}_{List}(q, \pi_q; \theta) = \sum_{i=1}^n \alpha(i) \left( -\tilde{P}(id^{(i)} | q; \theta) + \log \left( \sum_{k=i}^n \exp(\tilde{P}(id^{(k)} | q; \theta)) \right) \right)$$

## Dynamic scenarios

$$\begin{aligned}\mathcal{L}_{Global}(Q, \underline{\mathcal{D}}, I_D, I_Q; \theta) &= \mathcal{L}_{Indexing}(\underline{\mathcal{D}}, I_D; \theta) + \mathcal{L}_{Retrieval}(Q, I_Q; \theta) \\ &= - \sum_{d \in \underline{\mathcal{D}}} \log P(id \mid d; \theta) - \sum_{q \in Q} \sum_{id^q \in I_Q} \log P(id^q \mid q; \theta)\end{aligned}$$

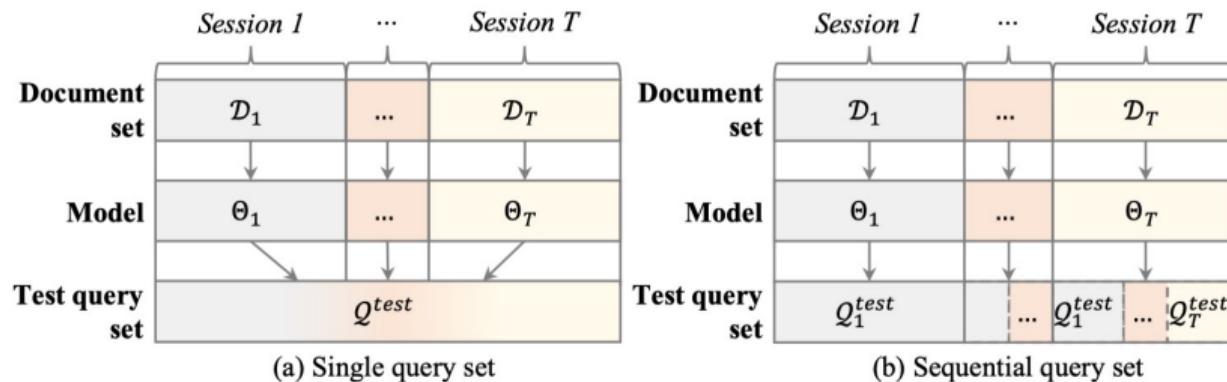
Information changes and new documents emerge incrementally over time

# Continual learning task: Formulation



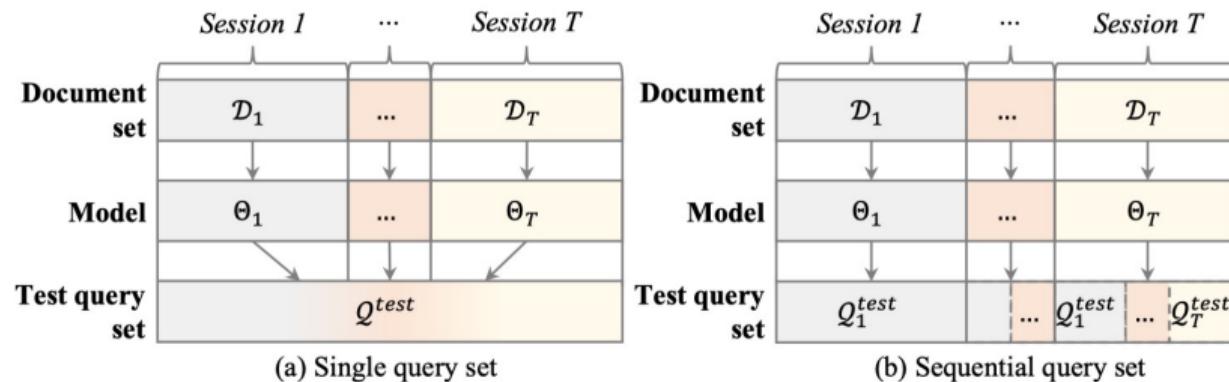
- **Initial model:** A large-scale base document set  $D_0$  and sufficiently many labeled query-document pairs

# Continual learning task: Formulation



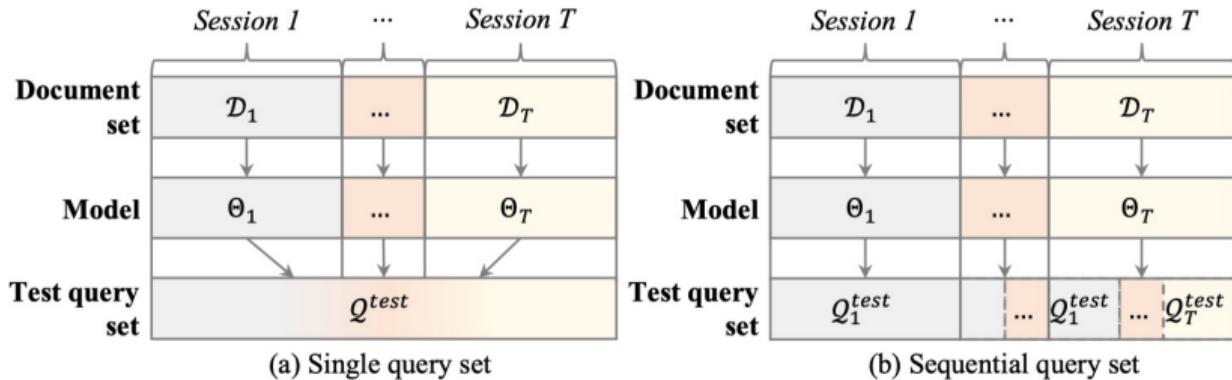
- **Initial model:** A large-scale base document set  $D_0$  and sufficiently many labeled query-document pairs
- **New datasets:**  $T$  new datasets  $D_1, \dots, D_T$ , from  $T$  sessions arriving in a sequential manner, which are only composed of newly encountered documents without queries related to these documents

# Continual learning task: Formulation



- **Initial model:** A large-scale base document set  $D_0$  and sufficiently many labeled query-document pairs
- **New datasets:**  $T$  new datasets  $D_1, \dots, D_T$ , from  $T$  sessions arriving in a sequential manner, which are only composed of newly encountered documents without queries related to these documents
- **Model update:** The new dataset  $D_t$  and previous datasets  $D_0, \dots, D_{t-1}$

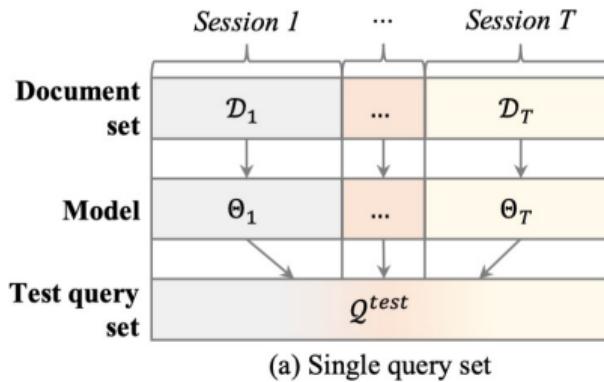
# Continual learning task: Evaluation



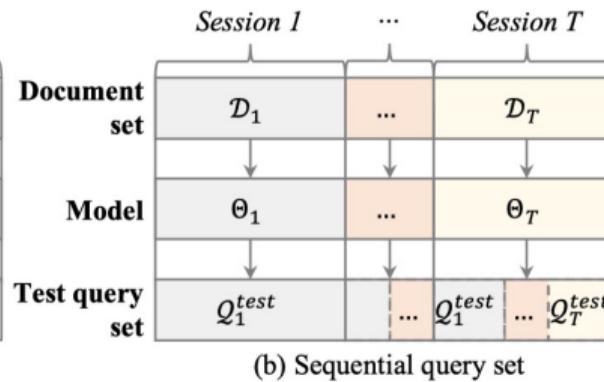
Two types of test query set for performance evaluation:

- **Single query set:** There is only one test query set, and their relevant documents arrive in different sessions
- **Sequential query set:** The test query set is specific for each session, and the relevant documents appear in existing sessions

# Continual learning task: Evaluation



(a) Single query set

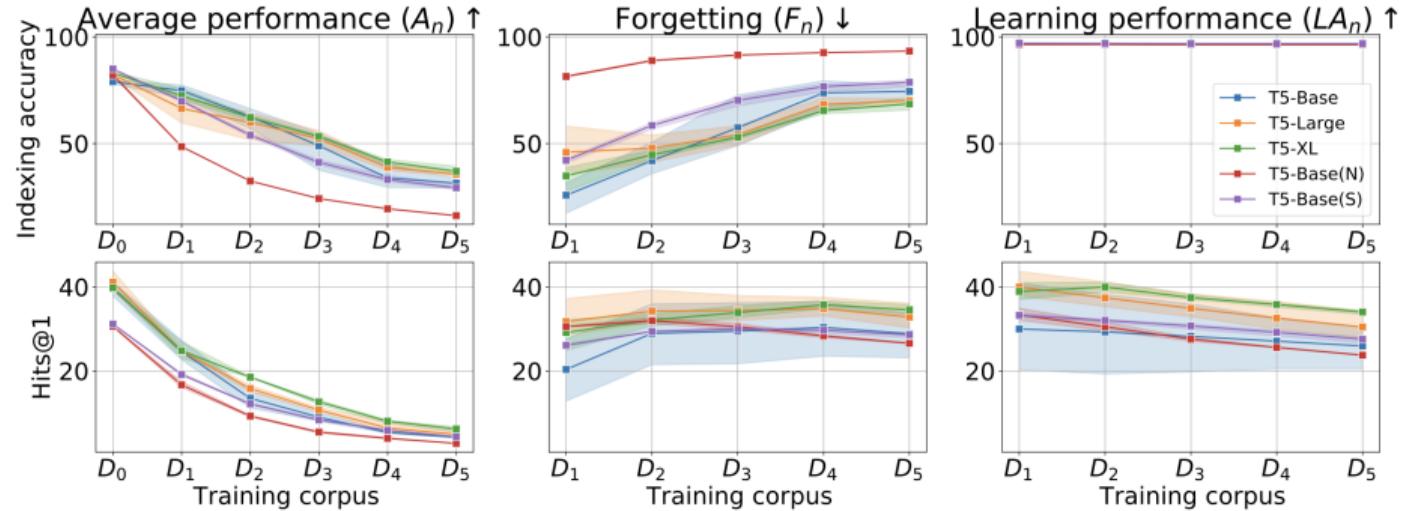


(b) Sequential query set

Two types of test query set for performance evaluation:

- **Single query set:** There is only one test query set, and their relevant documents arrive in different sessions
- **Sequential query set:** The test query set is specific for each session, and the relevant documents appear in existing sessions

# Catastrophic forgetting



The GR model undergoes **severe forgetting under continual indexing of new documents**

## Challenges of continual learning for GR

- How to incrementally index new documents with low computational and memory costs?

## Challenges of continual learning for GR

- How to incrementally index new documents with low computational and memory costs?
- How to prevent catastrophic forgetting for previously indexed documents and maintain the retrieval ability?

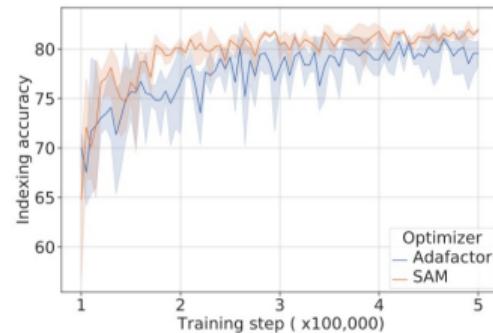
## DSI++ [Mehta et al., 2022]: Incrementally indexing new documents

- Docids: The new documents are assigned **unstructured atomic integers** as docids, and the GR model learns new embeddings for each of them

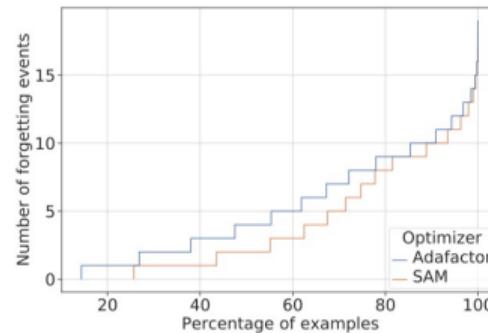
## DSI++ [Mehta et al., 2022]: Incrementally indexing new documents

- Docids: The new documents are assigned **unstructured atomic integers** as docids, and the GR model learns new embeddings for each of them
- **Modifying the training dynamics:** Since flatter minima implicitly alleviate forgetting, optimizing for flatter loss basins using Sharpness-Aware Minimization (SAM) as an objective allows the model to stably memorize more documents

# DSI++ [Mehta et al., 2022]: Incrementally indexing new documents



(a) Indexing accuracy during memorization



(b) Cumulative histogram of forgetting events

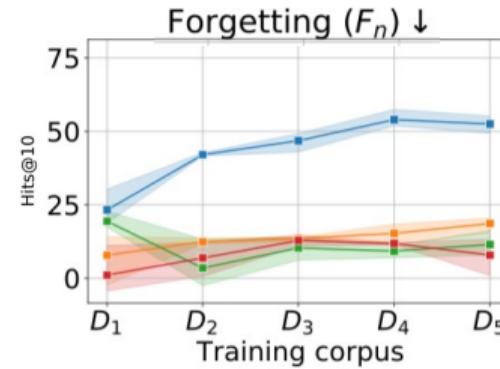
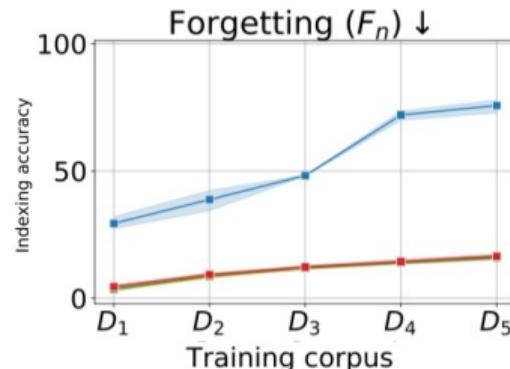
- SAM **outperforms Adafactor** in terms of the overall indexing accuracy
- SAM **undergoes less severe fluctuations** during the course of training

## DSI++ [Mehta et al., 2022]: Preventing catastrophic forgetting

- **Generative memory:** Train a query generator model to sample pseudo-queries for previously seen documents and supplement the query-docid pairs during continual indexing

## DSI++ [Mehta et al., 2022]: Preventing catastrophic forgetting

- **Generative memory:** Train a query generator model to sample pseudo-queries for previously seen documents and supplement the query-docid pairs during continual indexing
- It **reduces the forgetting**, and **improves average Hits@10 by +21.1%** over baselines



## Limitations of DSI++

- Learning embeddings for each individual new docid from scratch incurs prohibitively **high computational costs**

## Limitations of DSI++

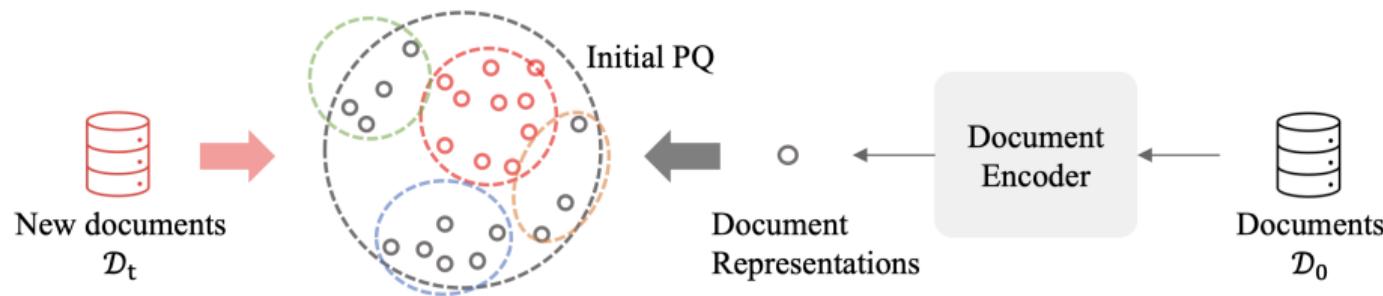
- Learning embeddings for each individual new docid from scratch incurs prohibitively **high computational costs**
- The relationships between new and old documents may not be easily obtained from **randomly-selected exemplars**

## CLEVER [Chen et al., 2023a]: Incrementally indexing new documents

Incremental product quantization (PQ) codes as identifiers: Update a partial quantization codebook according to two adaptive thresholds

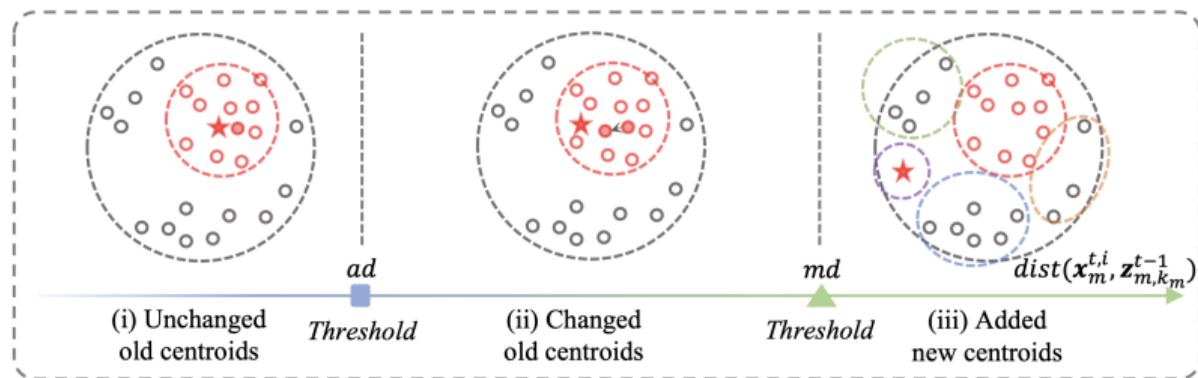
## CLEVER [Chen et al., 2023a]: Incrementally indexing new documents

Incremental product quantization (PQ) codes as identifiers: Update a partial quantization codebook according to two adaptive thresholds



- Build base PQ
  - Centroids are obtained via clustering over document representations
  - Document representations are learned with a bootstrapped training process

## CLEVER [Chen et al., 2023a]: Incremental product quantization



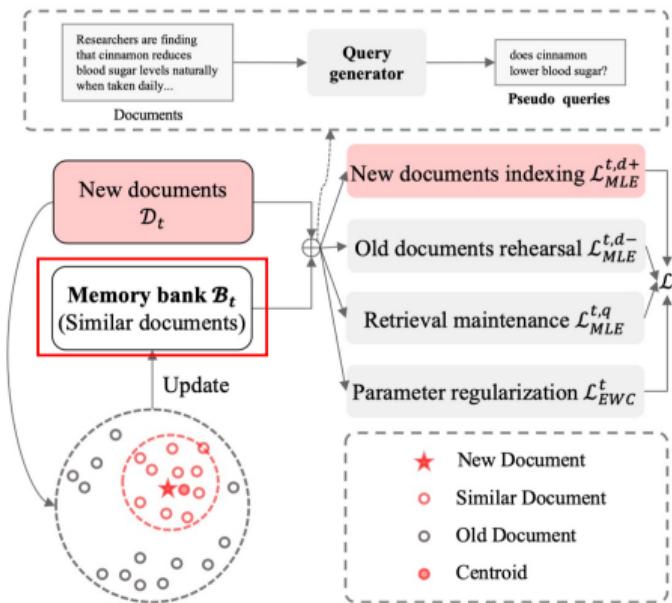
- Update adaptively
  - Dynamic thresholds: Average distance ( $ad$ ); maximum distance ( $md$ )
  - Three types of update for centroid representation: Depend on contributions to centroid update

## CLEVER [Chen et al., 2023a]: Preventing catastrophic forgetting

Memory-augmented learning mechanism: Form meaningful connections between old and new documents

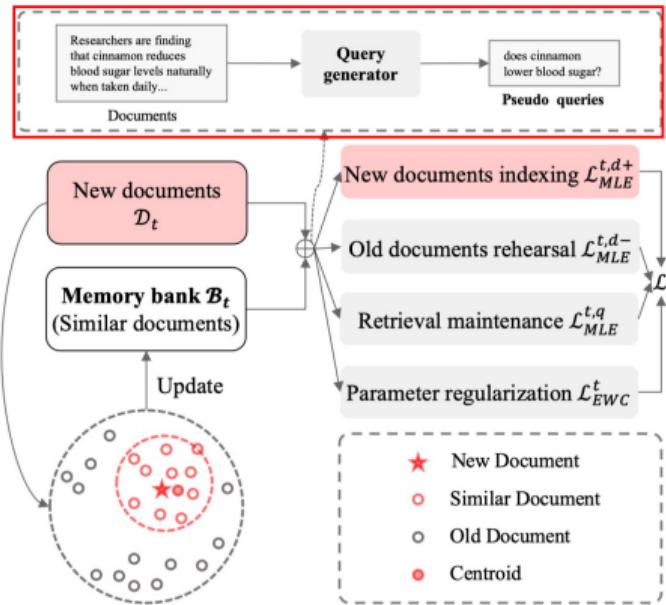
# CLEVER [Chen et al., 2023a]: Preventing catastrophic forgetting

Memory-augmented learning mechanism: Form meaningful connections between old and new documents



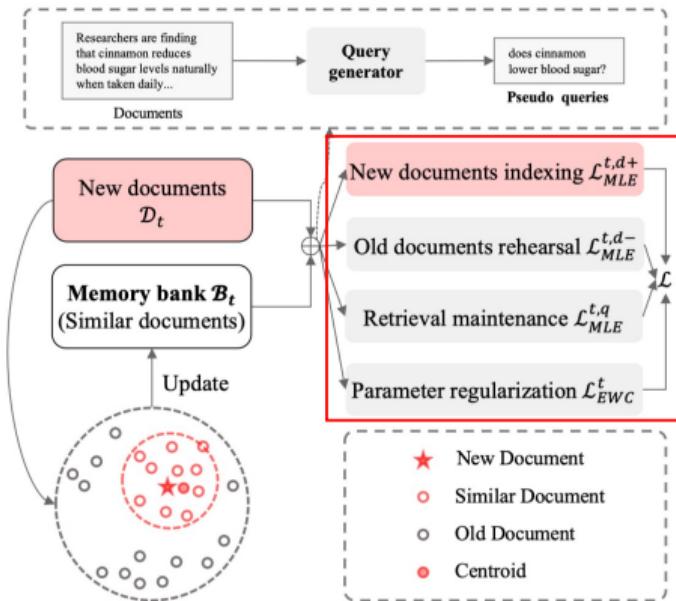
- **Dynamic memory bank:** Construct a memory bank with **similar documents** for each new session and replay the process of indexing them alongside the indexing of new documents

# CLEVER [Chen et al., 2023a]: Memory-augmented learning mechanism



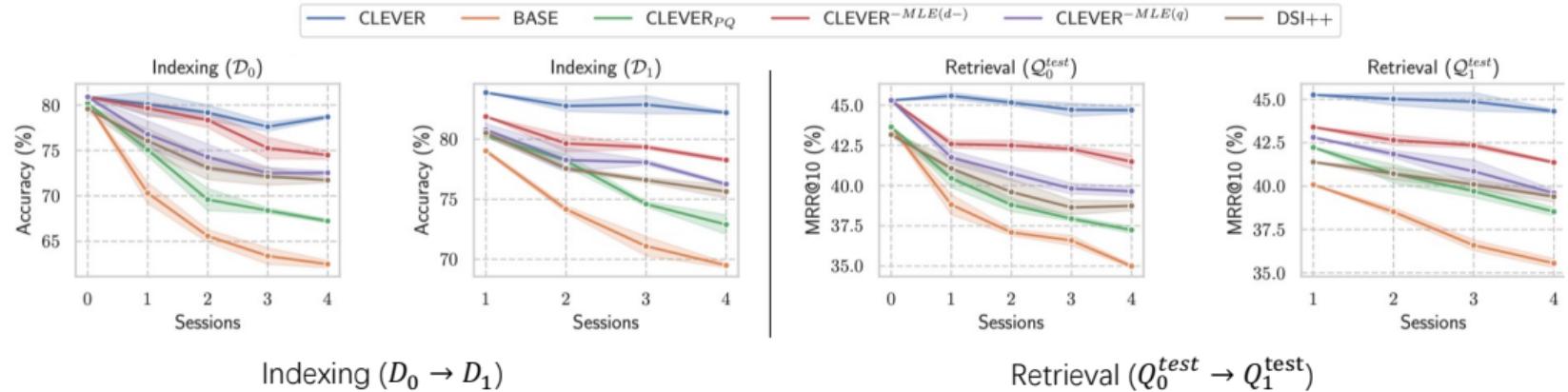
- **Pseudo query-docid pairs:** Train a query generator model to sample pseudo-queries for documents and **supplement the query-docid pairs** during indexing

# CLEVER [Chen et al., 2023a]: Memory-augmented learning mechanism



- **Sequentially training:** new documents indexing, old document rehearsal, retrieval maintenance losses and an **elastic weight consolidation** (EWC) loss as a regularization term

# CLEVER [Chen et al., 2023a]: Performance



- CLEVER almost avoids catastrophic forgetting on both indexing and retrieval tasks, showing its effectiveness in a dynamic setting

# A look back

Training approaches			
Standard approach (Tay et al. 2022)		- Simple	- Moderate performance
Stationary	Multi-granularity enhanced (Tang et al. 2023a)	- Enhancing the memorization ability	- Requiring extra tools for selecting important paragraphs or sentences
	Pseudo query enhanced (Zhuang et al. 2023)	- Reducing the gap between training and inference	- Depending on labeled data
	Pre-training based (Chen et al. 2022b)	- Addressing the issue of no or limited labeled data	- Depending on the quality of pre-training corpora and task design
Dynamic	DSI++ (Mehta et al. 2022)	Unstructured atomic integers	- Simple design
		Experience replay	- Good performance
	CLEVER (Chen et al. 2023a)	Incremental product quantization	- High efficiency
		Memory-augmented learning	- Better at capturing the relationship between old and new corpora - Better performance
			- Extra memory bank

## Revisit: Challenges of training approaches

- How to memorize the whole corpus effectively and efficiently?

## Revisit: Challenges of training approaches

- **How to memorize the whole corpus effectively and efficiently?**
  - Multi-granularity enhanced document content
  - Pre-training
  - Listwise optimization

## Revisit: Challenges of training approaches

- **How to memorize the whole corpus effectively and efficiently?**
  - Multi-granularity enhanced document content
  - Pre-training
  - Listwise optimization
- **How to learn heterogeneous tasks well within a single model?**

## Revisit: Challenges of training approaches

- **How to memorize the whole corpus effectively and efficiently?**
  - Multi-granularity enhanced document content
  - Pre-training
  - Listwise optimization
- **How to learn heterogeneous tasks well within a single model?**
  - Pseudo query enhanced input

## Revisit: Challenges of training approaches

- **How to memorize the whole corpus effectively and efficiently?**
  - Multi-granularity enhanced document content
  - Pre-training
  - Listwise optimization
- **How to learn heterogeneous tasks well within a single model?**
  - Pseudo query enhanced input
- **How to handle a dynamically evolving document collection?**

## Revisit: Challenges of training approaches

- **How to memorize the whole corpus effectively and efficiently?**
  - Multi-granularity enhanced document content
  - Pre-training
  - Listwise optimization
- **How to learn heterogeneous tasks well within a single model?**
  - Pseudo query enhanced input
- **How to handle a dynamically evolving document collection?**
  - Low computational and memory costs
  - Maintaining the retrieval ability

## Limitations

- Binary relevance: Current relevance modeling mainly relies on MLE, struggling with **real-world annotation relevance**

## Limitations

- Binary relevance: Current relevance modeling mainly relies on MLE, struggling with **real-world annotation relevance**
- Pointwise optimization: MLE is essentially a pointwise approach, and it is considered sub-optimal due to its disregard for the fundamental principle that ranking involves making predictions about **lists**

## Limitations

- Binary relevance: Current relevance modeling mainly relies on MLE, struggling with **real-world annotation relevance**
- Pointwise optimization: MLE is essentially a pointwise approach, and it is considered sub-optimal due to its disregard for the fundamental principle that ranking involves making predictions about **lists**
- **Capacity**: The relationships between docids and training approaches have not been systematically explored

## Limitations

- Binary relevance: Current relevance modeling mainly relies on MLE, struggling with **real-world annotation relevance**
- Pointwise optimization: MLE is essentially a pointwise approach, and it is considered sub-optimal due to its disregard for the fundamental principle that ranking involves making predictions about **lists**
- **Capacity**: The relationships between docids and training approaches have not been systematically explored

Model inference → **Section 5!**

## **Section 5: Inference strategies**

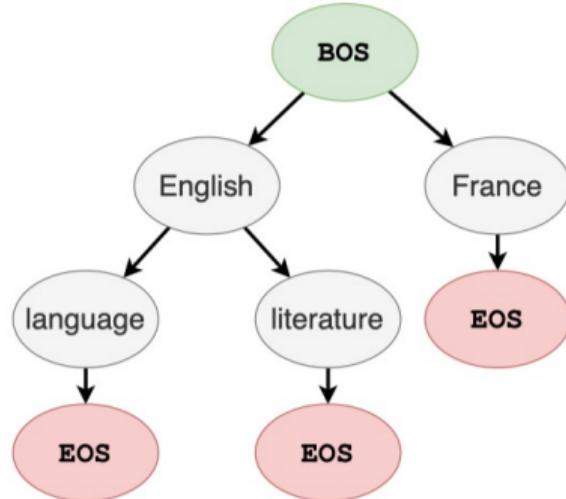
## Roadmap of inference strategies

- A **single identifier** to represent a document:
  - Constrained beam search with a prefix tree
  - Constrained greedy search with the inverted index

## Roadmap of inference strategies

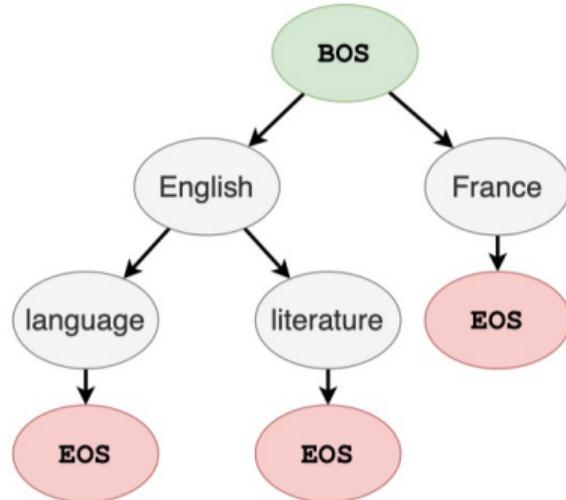
- A **single identifier** to represent a document:
  - Constrained beam search with a prefix tree
  - Constrained greedy search with the inverted index
- **Multiple identifiers** to represent a document
  - Constrained beam search with the FM-index
  - Scoring functions to aggregate the contributions of several identifiers

## Single identifier: Constrained beam search with a prefix tree



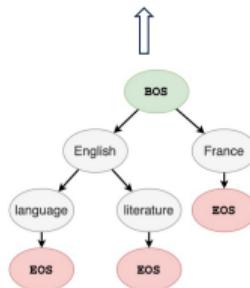
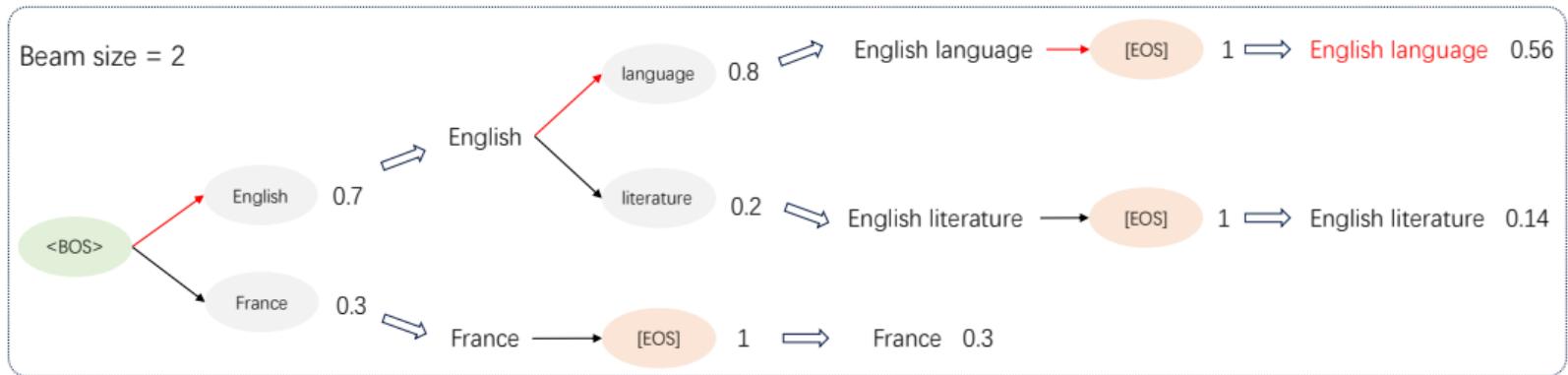
- For docids **considering order of tokens**
- **Applicable docids:** Naively structured strings, semantically structured strings, product quantization strings, titles, n-grams, URLs and pseudo queries

## Single identifier: Constrained beam search with a prefix tree



- For docids **considering order of tokens**
- **Applicable docids:** Naively structured strings, semantically structured strings, product quantization strings, titles, n-grams, URLs and pseudo queries
- Prefix tree: Nodes are annotated with tokens from the predefined candidate set. For each node, its children indicate all the allowed continuations from the prefix defined traversing the tree from the root to it

## Example



## Single identifier: Constrained greedy search with the inverted index

- Applicable docids: Important terms

## Single identifier: Constrained greedy search with the inverted index

- Applicable docids: Important terms
- Inverted index table: Enable the generation in **any permutations** (unordered docids) are constructed

## Single identifier: Constrained greedy search with the inverted index

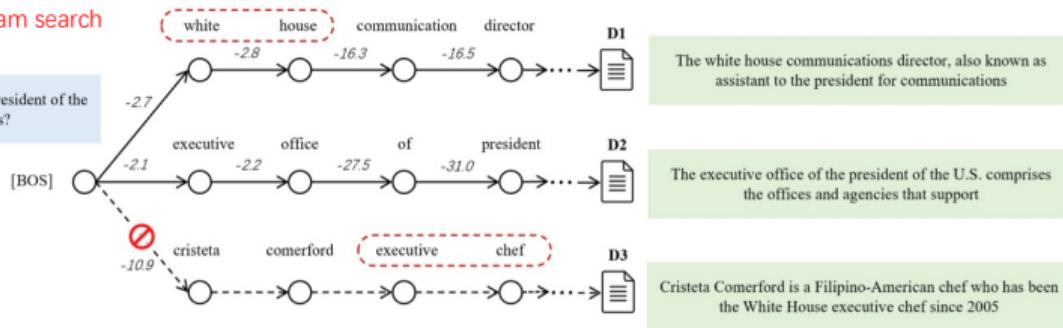
- Applicable docids: Important terms
- Inverted index table: Enable the generation in **any permutations** (unordered docids) are constructed
- Generation process: The model is expected to produce docids of the **highest generation likelihood**. At each step of generation, the terms from the inverted index table which give rise to the top-K generation likelihood are **greedily** selected

# Constrained beam search vs. Constrained greedy search

## Constrained beam search

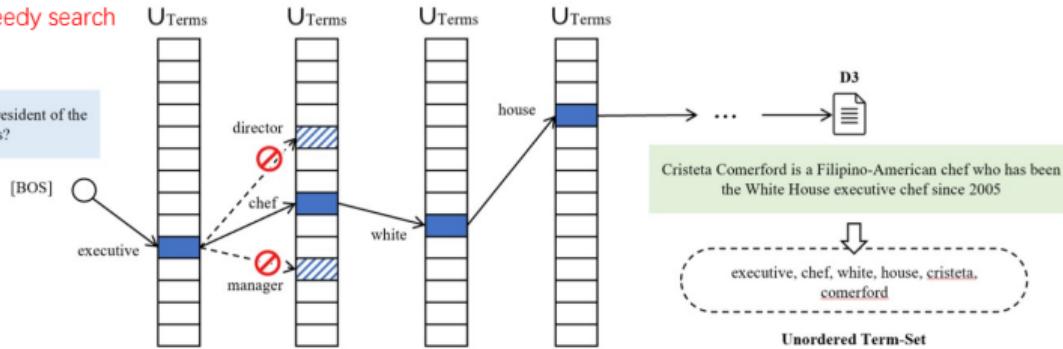
Beam Size=2

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

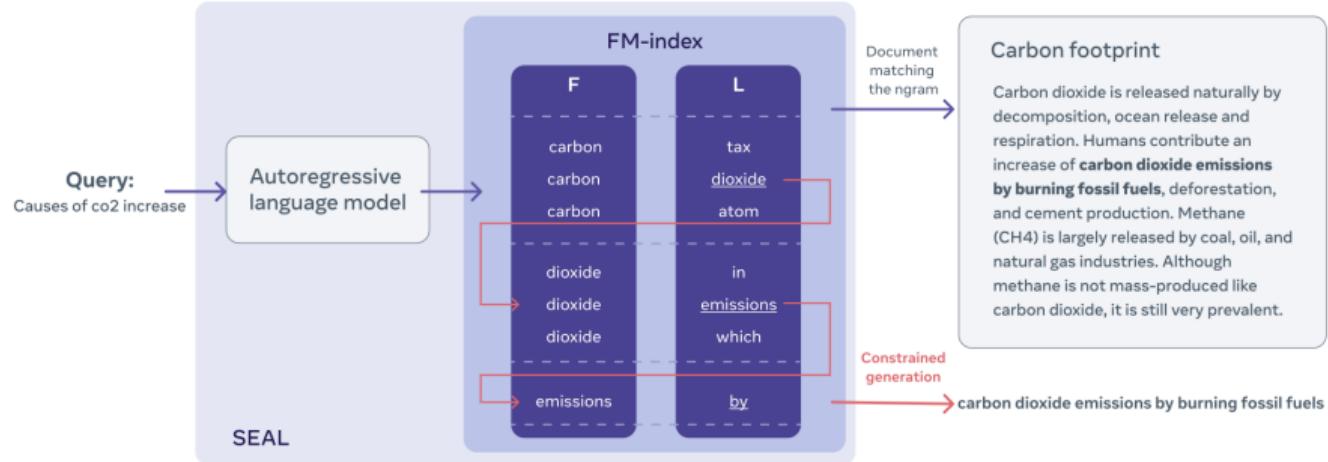


## Constrained greedy search

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

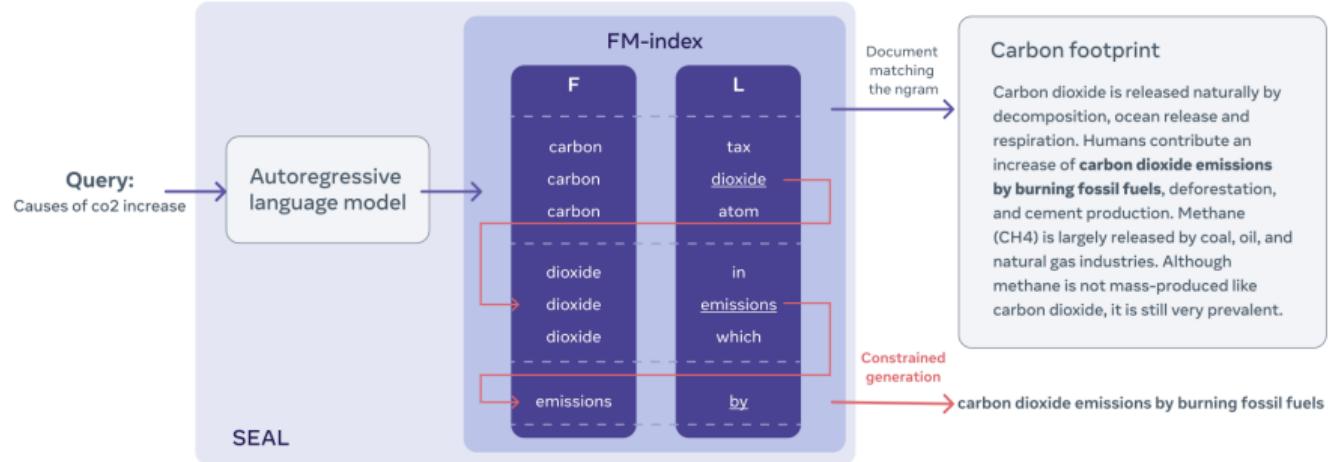


# Multiple identifiers: Constrained beam search with the FM-index



- Applicable docids: N-grams based docids

# Multiple identifiers: Constrained beam search with the FM-index



- Applicable docids: N-grams based docids
- FM-index: An index combining the Burrows-Wheeler Transform (BWT) with a few **small auxiliary data structures**

"Autoregressive Search Engines: Generating Substrings as Document Identifiers". Bevilacqua et al. [2022]

## FM-index

- **F** and **L**: **F** is an array of runs and **L** is the string's BWT
- Property: The relative rank of **F** and **L** stays the same; an FM-index can identify the list of possible token successors with constrained beam search

## FM-index: Decoding process

F	C	A	B	A	L
\$ <sup>6</sup>	C	A	B	A	C <sup>5</sup>
A <sup>2</sup>	B	A	C	\$	C <sup>1</sup>
A <sup>4</sup>	C	\$	C	A	B <sup>3</sup>
B <sup>3</sup>	A	C	\$	C	A <sup>2</sup>
C <sup>5</sup>	\$	C	A	B	A <sup>4</sup>
C <sup>1</sup>	A	B	A	C	\$ <sup>6</sup>

- Given the starting token \$, F is first used to find the contiguous range of rows corresponding to the token

## FM-index: Decoding process

F					L
\$ <sup>6</sup>	C	A	B	A	C <sup>5</sup>
A <sup>2</sup>	B	A	C	\$	C <sup>1</sup>
A <sup>4</sup>	C	\$	C	A	B <sup>3</sup>
B <sup>3</sup>	A	C	\$	C	A <sup>2</sup>
C <sup>5</sup>	\$	C	A	B	A <sup>4</sup>
C <sup>1</sup>	A	B	A	C	\$ <sup>6</sup>

- Given the starting token \$, **F** is first used to find the contiguous range of rows corresponding to the token
- L** is then used to examine the same range of rows to obtain the list of the next valid tokens

## FM-index: Decoding process

F					L
\$ <sup>6</sup>	C	A	B	A	C <sup>5</sup>
A <sup>2</sup>	B	A	C	\$	C <sup>1</sup>
A <sup>4</sup>	C	\$	C	A	B <sup>3</sup>
B <sup>3</sup>	A	C	\$	C	A <sup>2</sup>
C <sup>5</sup>	\$	C	A	B	A <sup>4</sup>
C <sup>1</sup>	A	B	A	C	\$ <sup>6</sup>

1. Given the starting token \$, **F** is first used to find the contiguous range of rows corresponding to the token
2. **L** is then used to examine the same range of rows to obtain the list of the next valid tokens
3. The valid tokens in **F** are selected based on the same ranks in **L**.

## FM-index: Decoding process

F					L
\$ <sup>6</sup>	C	A	B	A	C <sup>5</sup>
A <sup>2</sup>	B	A	C	\$	C <sup>1</sup>
A <sup>4</sup>	C	\$	C	A	B <sup>3</sup>
B <sup>3</sup>	A	C	\$	C	A <sup>2</sup>
C <sup>5</sup>	\$	C	A	B	A <sup>4</sup>
C <sup>1</sup>	A	B	A	C	\$ <sup>6</sup>

- Given the starting token \$, **F** is first used to find the contiguous range of rows corresponding to the token
  - L** is then used to examine the same range of rows to obtain the list of the next valid tokens
  - The valid tokens in **F** are selected based on the same ranks in **L**.
- By iteratively repeating the above procedure, a valid n-gram with arbitrary size can be found

## FM-index: N-gram level scores

Given an input query  $q$ , we obtain the weight of each predicted n-gram  $n$ :

$$\text{score}(n, q) = \max(0, \log \frac{P(n|q)(1 - P(n))}{P(n)(1 - P(n|q))}),$$

where  $P(n|q)$  is the probability of the generative model decoding  $n$  conditioned on  $q$ , and  $p(n)$  denotes the unconditional n-gram probability.

## N-gram level to document level scores

How to **aggregate** the contribution of multiple generated n-gram identifiers to its corresponding documents?

## Aggregation functions: SEAL [Bevilacqua et al., 2022]

The document-level rank score combines the n-gram level rank score  $score(n, q)$  and coverage weight  $cover(n, K)$ :

$$score(d, q) = \sum_{n \in K^d} score(n, q)^\alpha \times cover(n, K),$$

where  $K$  denotes all the generated n-grams,  $K^d$  is the subset of n-grams in  $K$  that appear in  $d$ ,  $\alpha$  is a hyperparameter

## Aggregation functions: SEAL [Bevilacqua et al., 2022]

For **docid repetition** problem

- Coverage weight  $cover(n, K)$ : Avoid the overscoring of very repetitive documents, where many similar n-grams are matched

$$cover(n, K) = 1 - \beta + \beta \frac{|set(n) \setminus C(n, K)|}{|set(n)|},$$

where  $\beta$  is a hyperparameter,  $set(n)$  is the set of tokens in  $n$ , and  $C(n, K)$  is the union of all tokens in  $K$  with top- $g$  highest scores

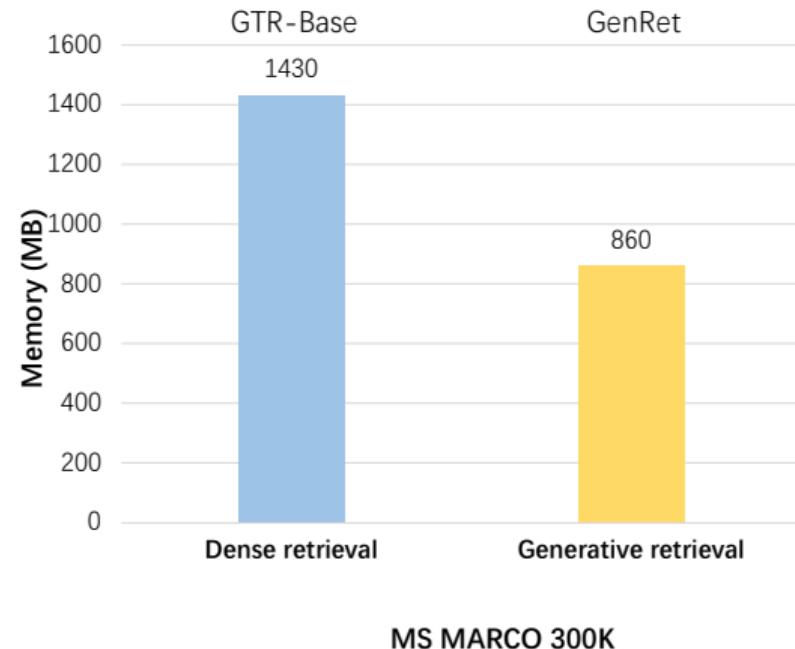
## Aggregation functions: MINDER [Li et al., 2023]

The document-level rank score: Sum of the scores of its covered docid

$$score(q, d) = \sum_{i_d \in I_d} P(i_d | q),$$

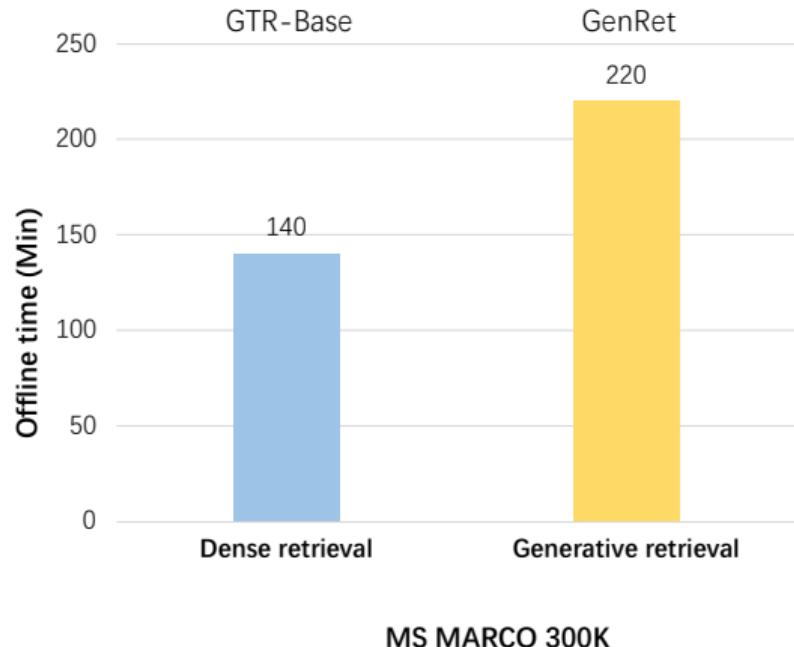
where  $P(i_d | q)$  is the generated likelihood score of the docid  $i_d$  of the document  $d$ . And  $I_d$  denotes the docids generated for  $d$

## Inference efficiency: Memory footprint



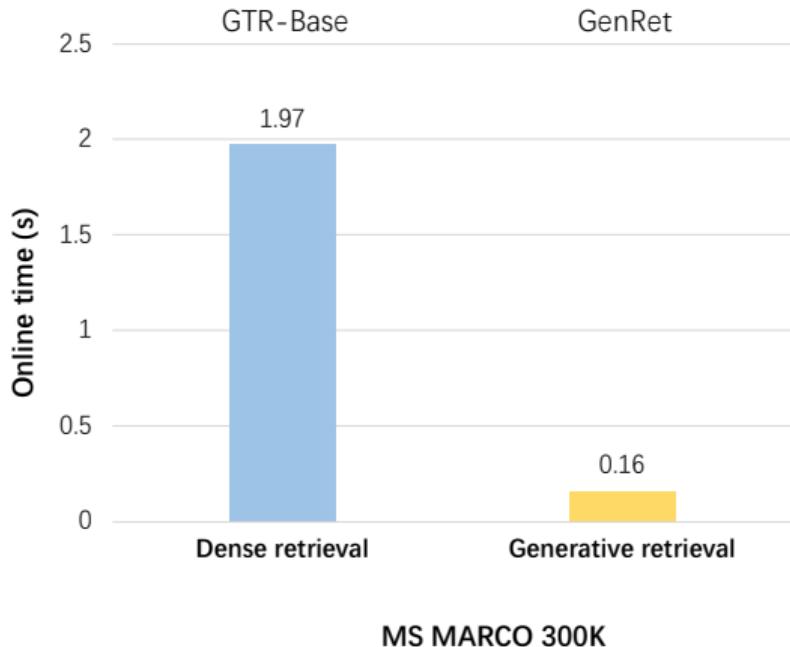
- The memory footprint of the GR model GenRet is **smaller** than that of the traditional dense retrieval method GTR, e.g., 1.6 times

## Inference efficiency: Offline latency



- GenRet takes a longer time for offline indexing, as the use of auxiliary models. GTR's offline time consumption comes from document encoding

## Inference efficiency: Online latency



- Compared with the traditional dense retrieval model GTR, the GR model GenRet is **faster**, e.g., 12 times

# A look back

Inference strategies			
A single docid	Constrained beam search with prefix tree (De Cao et al. 2021)	- Simple	- It cannot generate in an unordered manner
	Constrained greedy search with inverted index (Zhang et al. 2023)	- It can generate in any permutations of docids	- It may require handling a significant amount of duplicate terms
Multiple docids	Constrained beam search with FM-index (Bevilacqua et al. 2022)	- It can store all the information of documents - The contributions of multiple docids comprehensively are considered	- It cannot generate in an unordered manner - Complex construction - Complex aggregation functions
	Scoring functions (Li et al. 2023)	- The contributions of multiple docids comprehensively are considered - Simple aggregation functions	- Depending on design

## Revisit: Challenges of model inference

- How to generate valid docids?

## Revisit: Challenges of model inference

- **How to generate valid docids?**
  - Constrained generation mechanism based on prefix tree, inverted index table or FM-index

## Revisit: Challenges of model inference

- **How to generate valid docids?**
  - Constrained generation mechanism based on prefix tree, inverted index table or FM-index
- **How to organize the docids for large scale corpus?**

## Revisit: Challenges of model inference

- **How to generate valid docids?**
  - Constrained generation mechanism based on prefix tree, inverted index table or FM-index
- **How to organize the docids for large scale corpus?**
  - Exploiting the structured docid space

## Revisit: Challenges of model inference

- **How to generate valid docids?**
  - Constrained generation mechanism based on prefix tree, inverted index table or FM-index
- **How to organize the docids for large scale corpus?**
  - Exploiting the structured docid space
- **How to generate a ranked list of docids for a query?**

## Revisit: Challenges of model inference

- **How to generate valid docids?**
  - Constrained generation mechanism based on prefix tree, inverted index table or FM-index
- **How to organize the docids for large scale corpus?**
  - Exploiting the structured docid space
- **How to generate a ranked list of docids for a query?**
  - One-by-one generation based on likelihood probabilities

## Limitations

- **One-by-one generation:** There are limited works on one-time generation, i.e., directly decoding a sequence of docids

## Limitations

- **One-by-one generation:** There are limited works on one-time generation, i.e., directly decoding a sequence of docids
- **Dependency on prior knowledge:** Some heuristic scoring functions may require prior knowledge or manually designed rules

## Limitations

- **One-by-one generation:** There are limited works on one-time generation, i.e., directly decoding a sequence of docids
- **Dependency on prior knowledge:** Some heuristic scoring functions may require prior knowledge or manually designed rules
- **Not universally applicable:** Different tasks and scenarios may require different structures to handle decoding constraints

## Limitations

- **One-by-one generation:** There are limited works on one-time generation, i.e., directly decoding a sequence of docids
- **Dependency on prior knowledge:** Some heuristic scoring functions may require prior knowledge or manually designed rules
- **Not universally applicable:** Different tasks and scenarios may require different structures to handle decoding constraints
- **Large-scale corpus:** Current docid space is limited to millions of documents

## Limitations

- **One-by-one generation:** There are limited works on one-time generation, i.e., directly decoding a sequence of docids
- **Dependency on prior knowledge:** Some heuristic scoring functions may require prior knowledge or manually designed rules
- **Not universally applicable:** Different tasks and scenarios may require different structures to handle decoding constraints
- **Large-scale corpus:** Current docid space is limited to millions of documents

Applications → **Section 6!**

## **Section 6: Applications**

# A range of target tasks

## Fact Verification

De Cao et al. 2021, Chen et al. 2022b,  
Chen et al. 2022a, Thorne et al. 2022,  
Lee et al. 2023

## Open Domain QA

De Cao et al. 2021, Chen et al. 2022b,  
Zhou et al. 2022, Lee et al. 2023

## Entity Linking

De Cao et al. 2021, Chen et al. 2022b,  
Lee et al. 2023

Knowledge-intensive language tasks

# A range of target tasks

## Fact Verification

De Cao et al. 2021, Chen et al. 2022b,  
Chen et al. 2022a, Thorne et al. 2022,  
Lee et al. 2023

## Open Domain QA

De Cao et al. 2021, Chen et al. 2022b,  
Zhou et al. 2022, Lee et al. 2023

## Entity Linking

De Cao et al. 2021, Chen et al. 2022b,  
Lee et al. 2023

## Multi-hop retrieval

Lee et al. 2022

## Recommendation

Si et al. 2023, Rajput et al. 2023

## Code retrieval

Naddem et al. 2022

More retrieval tasks

# A range of target tasks

## Fact Verification

De Cao et al. 2021, Chen et al. 2022b,  
Chen et al. 2022a, Thorne et al. 2022,  
Lee et al. 2023

## Open Domain QA

De Cao et al. 2021, Chen et al. 2022b,  
Zhou et al. 2022, Lee et al. 2023

## Entity Linking

De Cao et al. 2021, Chen et al. 2022b,  
Lee et al. 2023

## Multi-hop retrieval

Lee et al. 2022

## Recommendation

Si et al. 2023, Rajput et al. 2023

## Code retrieval

Naddem et al. 2022

## Official site retrieval

Tang et al. 2023a

Industry retrieval tasks

## How to adapt a GR model for a task?

- Docid design
- Training approach
- Inference strategy

# Knowledge-intensive language tasks

## Slot Filling

INPUT:  
Star Trek [SEP] creator

OUTPUT:  
Gene Roddenberry

PROVENANCE:  
17157886-1

zsRE

## Open Domain QA

INPUT:  
When did Star Trek go off the air

OUTPUT:  
June 3, 1969

PROVENANCE:  
17157886-5

NQ

INPUT:  
Which Star Trek star directed Three Men and a Baby?

OUTPUT:  
Leonard Nimoy

PROVENANCE:  
17157886-4, 596639-7

TQA

INPUT:  
TrekLanta (formerly "TrekTrax Atlanta") is an annual convention for what American science fiction media franchise?

OUTPUT:  
Star Trek

PROVENANCE:  
17157886-1, 28789994-6

HoPo



Knowledge source:  
5.9 Million Wikipedia pages

## Star Trek 17157886

Star Trek is an American media franchise based on the science fiction television series created by Gene Roddenberry.<sup>1</sup> [...] It followed the interstellar adventures of Captain James T. Kirk (William Shatner) and his crew aboard the starship USS "Enterprise", a space exploration vessel built by the United Federation of Planets in the 23rd century.<sup>2</sup> The "Star Trek" canon includes "The Original Series", an animated series, five spin-off television series, the film franchise, and further adaptations in several media.<sup>3</sup> [...] The original 1966–69 series featured William Shatner as Captain James T. Kirk, Leonard Nimoy<sup>4</sup> as Spock, DeForest Kelley as Dr. Leonard "Bones" McCoy, James Doohan as Montgomery "Scotty" Scott, Nichelle Nichols as Uhura, George Takei as Hikaru Sulu, and Walter Koenig as Pavel Chekov. During the series' first run, it earned several nominations for the Hugo Award for Best Dramatic Presentation, and won twice. [...] NBC canceled the show after three seasons; the last original episode aired on June 3, 1969.<sup>5</sup> [...]

## Three Men and a Baby 596639

Three Men and a Baby is a 1987 American comedy film directed by Leonard Nimoy<sup>7</sup> and starring Tom Selleck, Steve Guttenberg, Ted Danson and Nancy Travis. [...]

## TrekLanta 28789994

TrekLanta is an annual "Star Trek" convention based in Atlanta, Georgia that places special emphasis on fan-based events, activities, programming and productions.<sup>6</sup> [...]

## Dialogue

INPUT:

I am a big fan of Star Trek, the American franchise created by Gene Roddenberry. I don't know much about it. When did the first episode air?

It debuted in 1966 and aired for 3 seasons on NBC.

What is the plot of the show?

OUTPUT:

William Shatner plays the role of Captain Kirk. He did a great job.

PROVENANCE:  
17157886-2

WoW

## Fact Checking

INPUT:

Star Trek had spin-off television series.

OUTPUT:

Supports

PROVENANCE:  
17157886-3

FEV

## Entity Linking

INPUT:

[...] Currently the site offers five movie collections ranging from \$149 for 10 [START\_ENT]Star Trek[END\_ENT] films to \$1,125 for the eclectic Movie Lovers' Collection of 75 movies. [...]

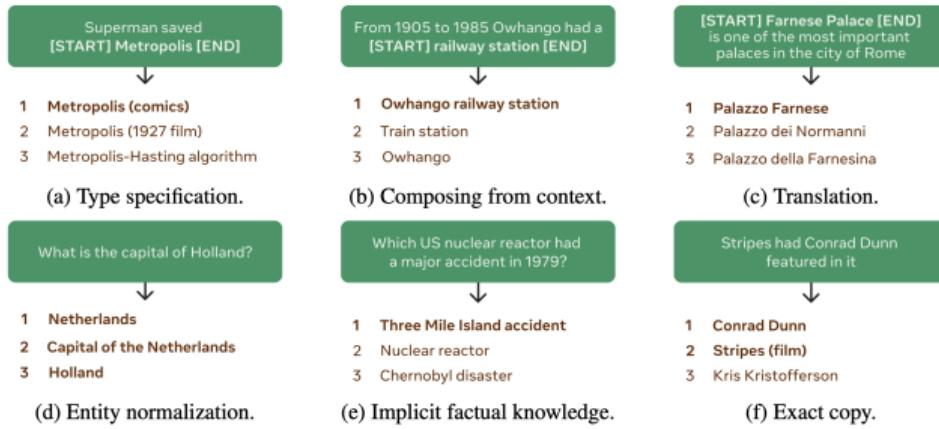
OUTPUT:

Star Trek

PROVENANCE:  
17157886

CnWn

# KILT example: GENRE [De Cao et al., 2021]

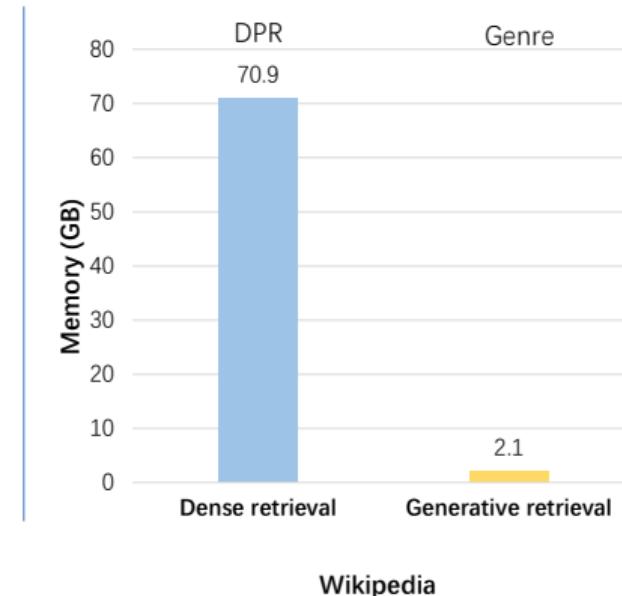
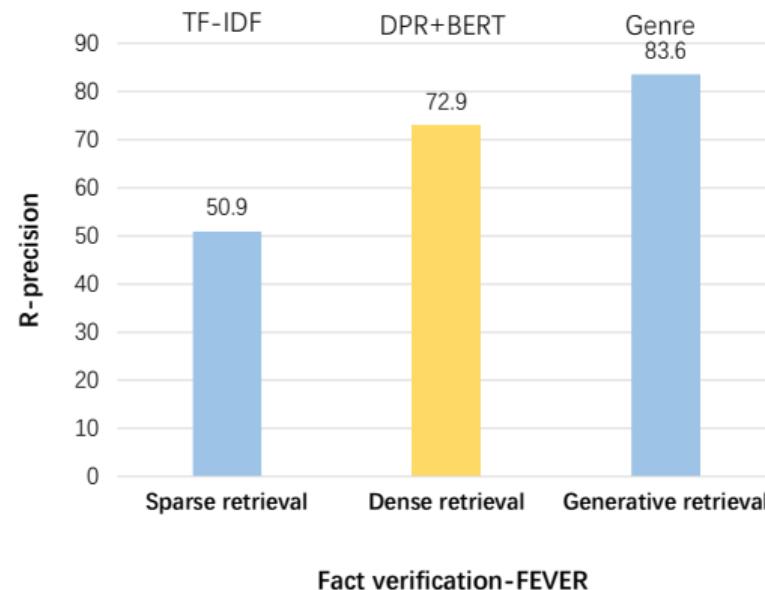


- Entity retrieval: Entity disambiguation, document retrieval, and etc
- Corpus: Wikipedia
- Input: Query
- Output: Destination/ relevant pages' title

## KILT example: GENRE [De Cao et al., 2021]

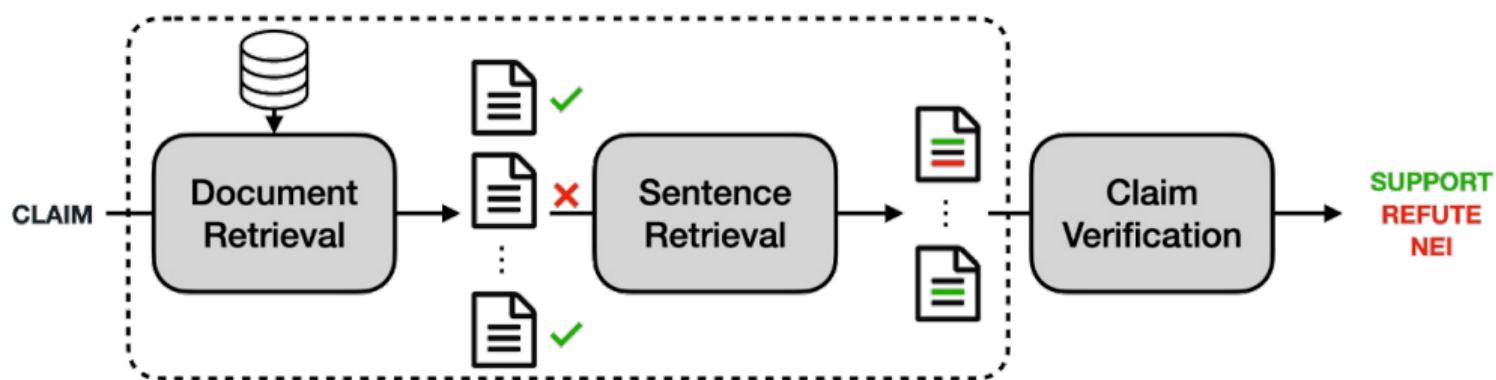
- **Docid:** Titles
- **Training:** MLE objective with document-title and query-title pairs
- **Inference:** Constrained beam search with a prefix tree

# KILT example: GENRE [De Cao et al., 2021]

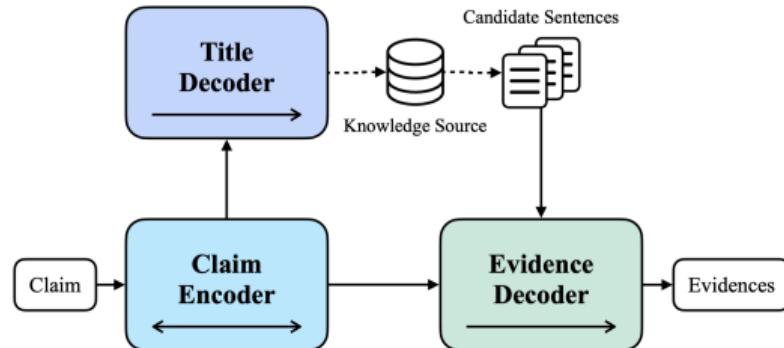


## KILT example: GERE [Chen et al., 2022a]

- Fact verification: Verify a claim using multiple evidential sentences from trustworthy corpora
  - Input: Claim
  - Output: Support/Refute/Not enough information

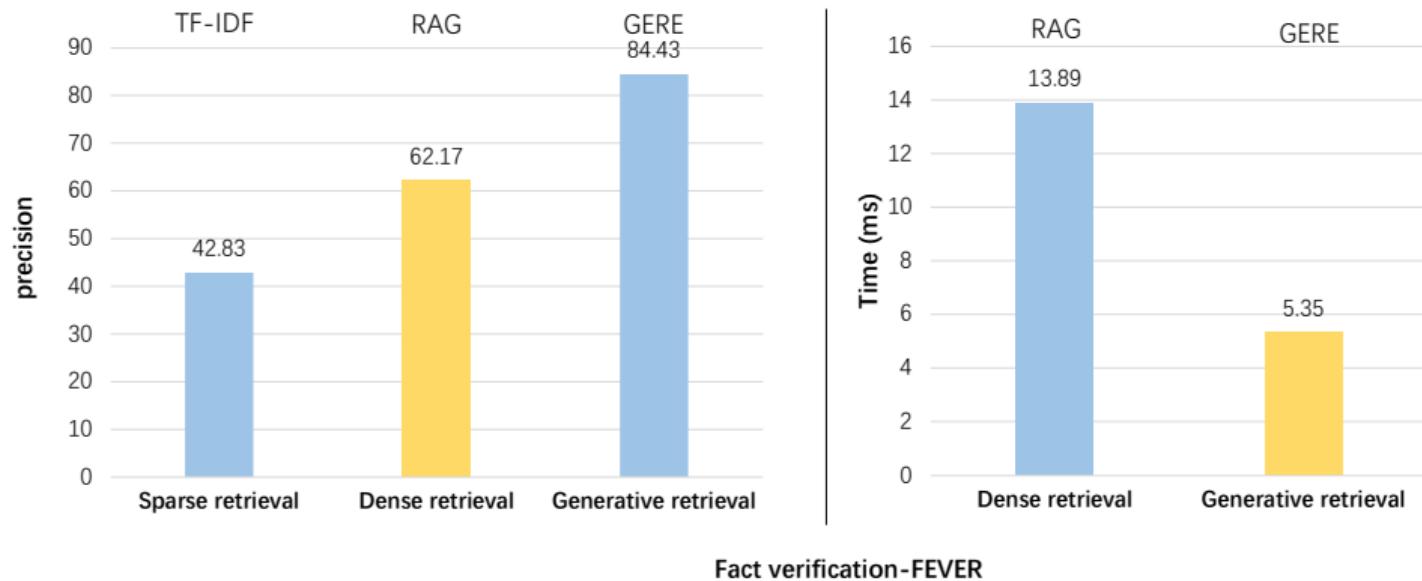


## KILT example: GERE [Chen et al., 2022a]



- **Docid:** Titles
- **Training:** MLE objective with claim-title and claim-evidence pairs
- **Inference:** Constrained beam search with a prefix tree

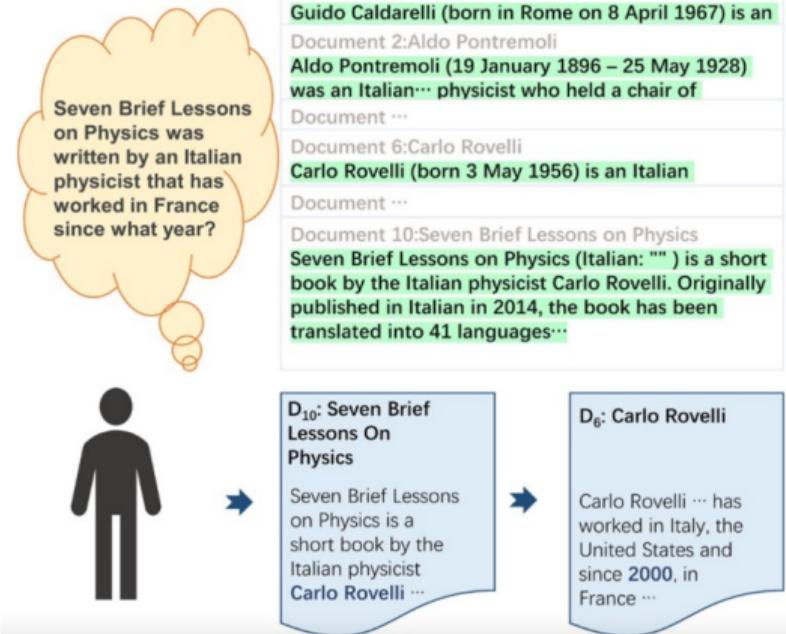
## KILT example: GERE [Chen et al., 2022a]



"GERE: Generative evidence retrieval for fact verification". [Chen et al. \[2022a\]](#)

# Multi-hop retrieval [Lee et al., 2022]

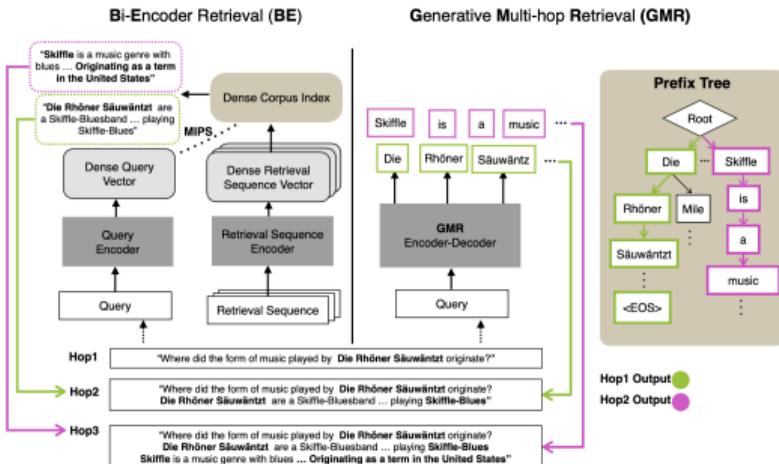
Image source: Memory enhances ChatGPT performance in multi-hop QA



- Multi-hop retrieval

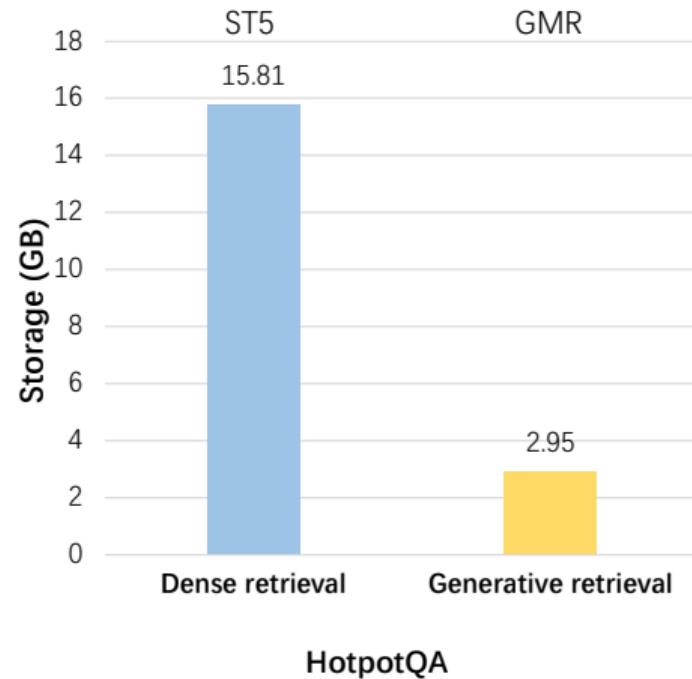
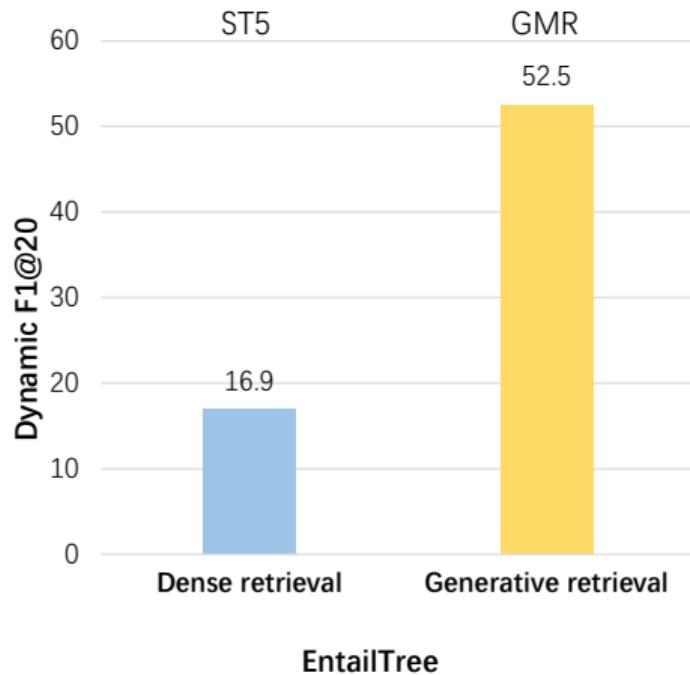
- One needs to retrieve multiple documents that together provide sufficient evidence to answer the query
- Previously retrieved items are appended to the query while iterating through multiple hops

# Multi-hop retrieval [Lee et al., 2022]



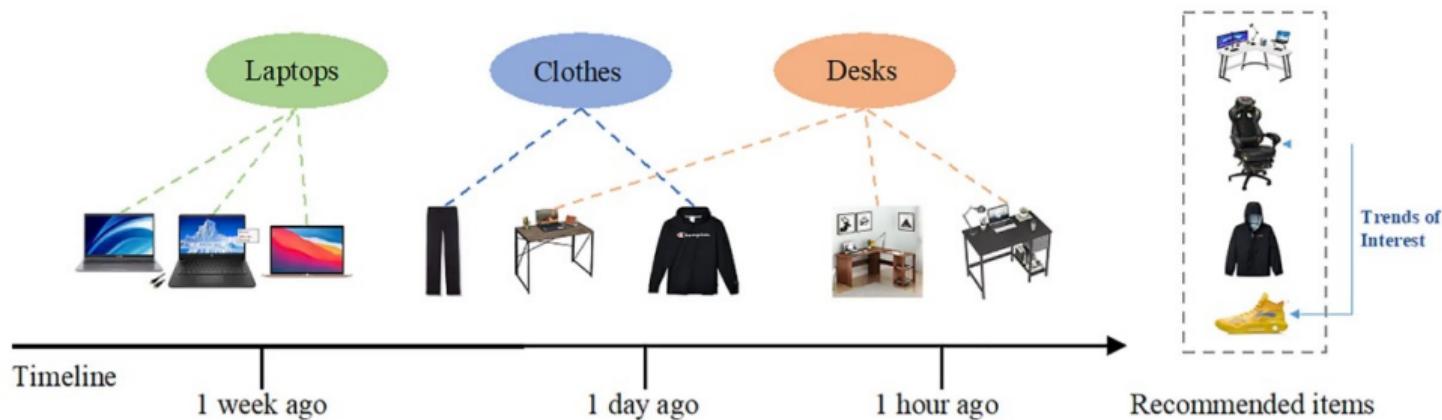
- **Docid:** Word-based answer
- **Jointly training:**
  - Indexing: Randomly select the first  $m$  words of the document as input and predict the remaining words with MLE
  - Retrieval: Learn pseudo query-answer pairs with MLE
- **Inference:** Constrained beam search with a prefix tree

## Multi-hop retrieval [Lee et al., 2022]

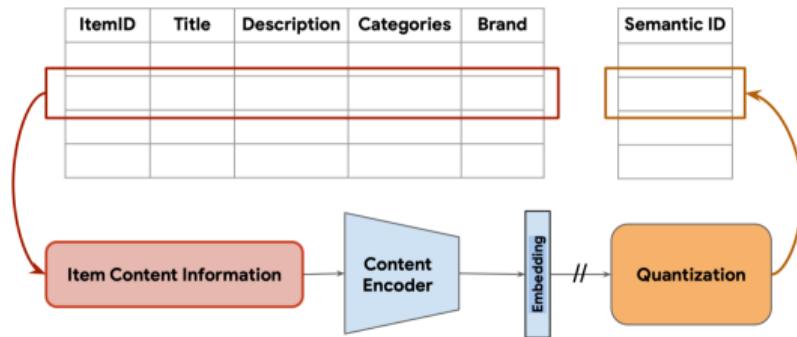


# Recommendation [Rajput et al., 2023]

- Sequential recommendation: Help users discover content of interest and are ubiquitous in various recommendation domains
  - Input: User history
  - Output: Next item identifier

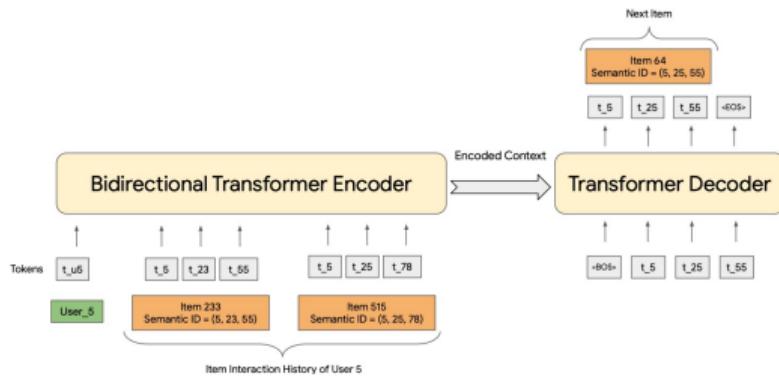


## Recommendation [Rajput et al., 2023]



- **Docid:** Product quantization strings
- **Docid training:** Train a residual-quantized variational autoencoder model with a docid reconstruction loss and a multi-stage quantization loss

# Recommendation [Rajput et al., 2023]

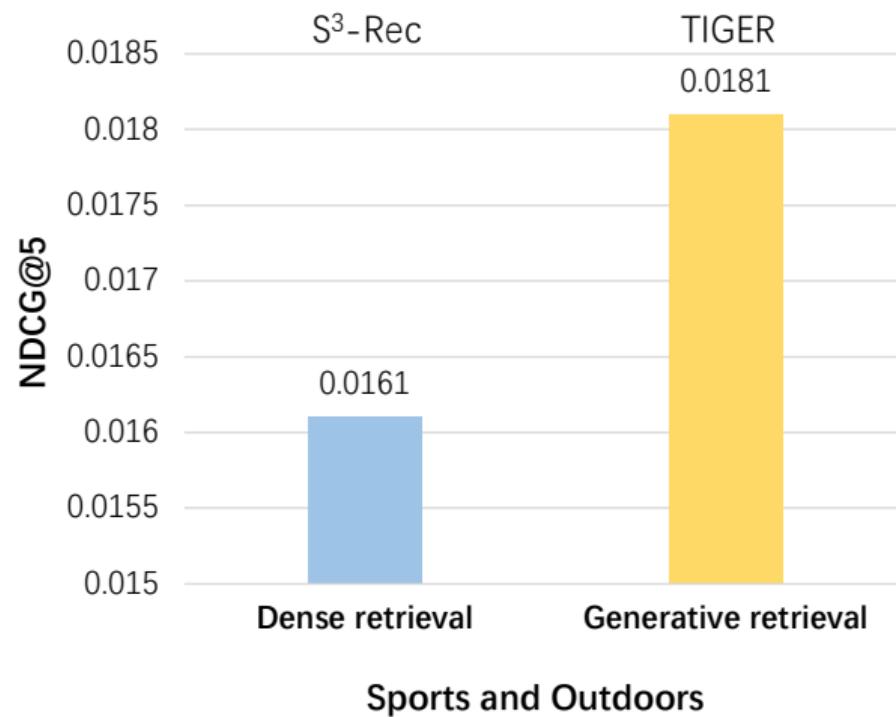


- **Recommendation training**

- Construct item sequences for every user by sorting chronologically the items they have interacted with
- Given item sequences, the recommender system's task is to predict the next item with MLE

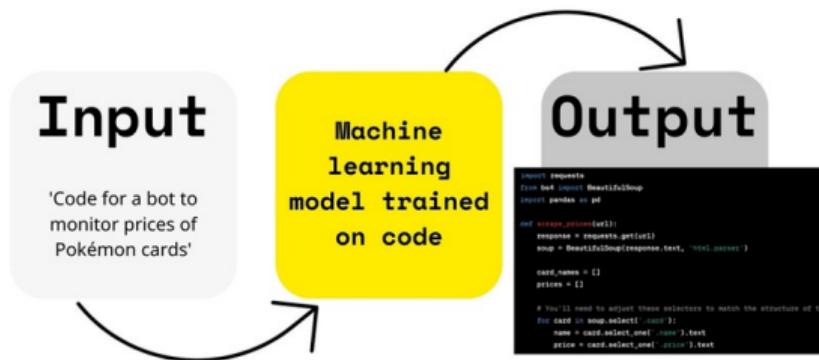
- **Inference:** Beam search

## Recommendation [Rajput et al., 2023]

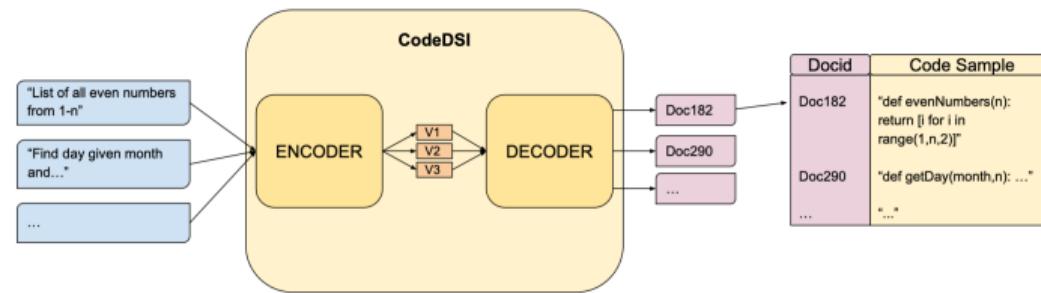


## Code retrieval [Nadeem et al., 2022]

- Code retrieval: A model takes natural language queries as input and, in turn, relevant code samples from a database are returned
  - Input: Query
  - Output: Relevant code samples

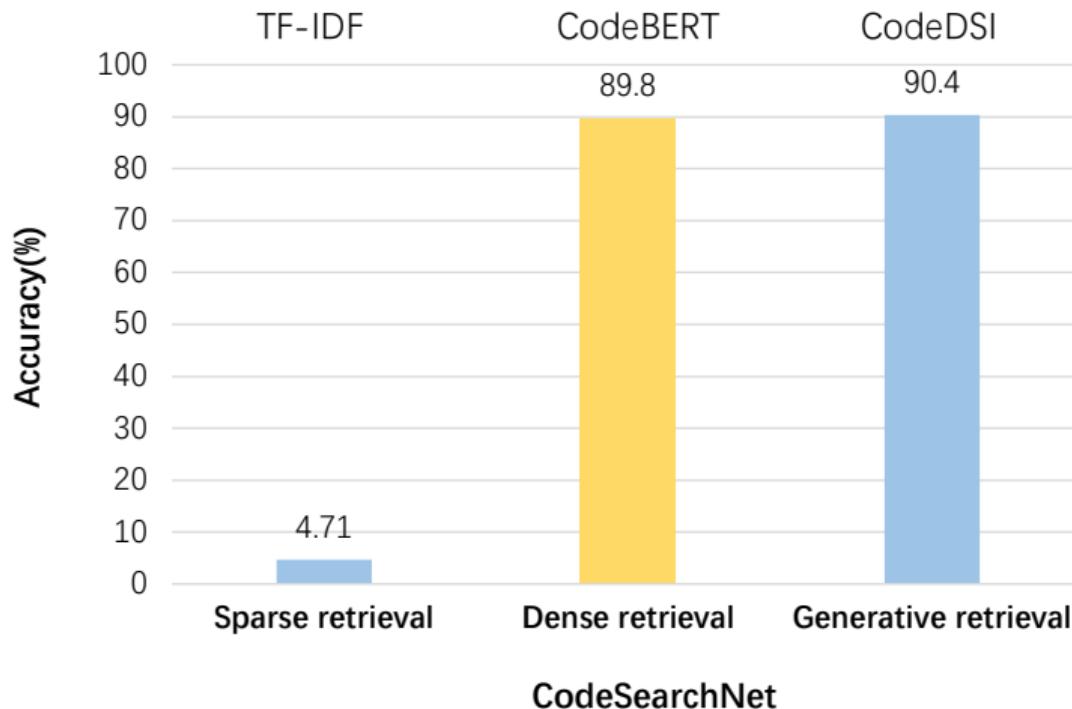


## Code retrieval [Nadeem et al., 2022]



- **Docid:** Naively structured strings/ semantically structured strings
- **Training:** Standard indexing loss with code-docid pairs and retrieval loss with query-docid pairs
- **Inference:** Beam search

## Code retrieval [Nadeem et al., 2022]

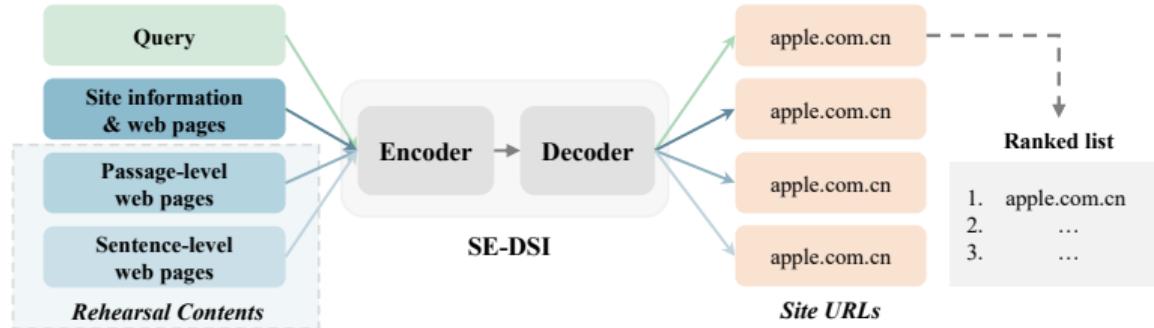


# Official site retrieval [Tang et al., 2023a]



- Official sites: Web pages that have been operated by universities, departments, or other administrative units

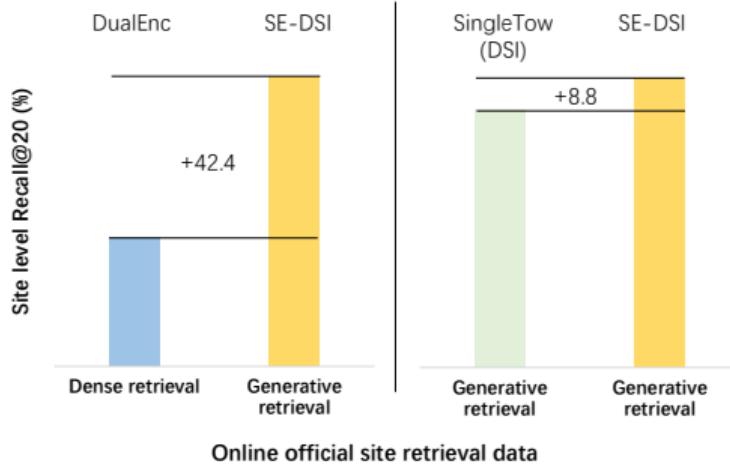
# Official site retrieval [Tang et al., 2023a]



- **Docid:** Unique site URLs
- **Jointly training:**
  - Indexing: Learn site information (site name/ site domain/ ICP record) - docid pairs, web pages-docid pairs, and important web pages-docid pairs with MLE
  - Retrieval: Learn query - docid pairs with MLE
- **Inference:** Constrained beam search with a prefix tree

# Official site retrieval [Tang et al., 2023a]

## Online retrieval performance:



## Inference comparison:

- Memory footprint: SE-DSI's memory is reduced by about 31 times compared to RepBERT
- Inference speed: SE-DSI's speed is significantly improved by about 2.5 times compared to RepBERT

## Applications: limitations

- Generalizing to **ultra-large-scale corpora** remains a challenge
- How to adapt to the significant **dynamic** changes in large-scale corpora for **online** applications

## **Section 7: Challenges & Opportunities**

## Tutorial summary

- **Definition & preliminaries**
- **Generative retrieval: docid design**
  - Single docids: number-based and word-based identifiers
  - Multiple docids: single type and diverse types
- **Generative retrieval: training approaches**
  - Stationary scenarios: supervised learning and pre-training
  - Dynamic scenarios
- **Generative retrieval: inference strategies**
  - Single docids: constrained greedy search, constrained beam search and FM-index
  - Multiple docids: aggregation functions
- **Generative retrieval: applications**

## Pros of generative retrieval

Information retrieval in the era of language models

## Pros of generative retrieval

Information retrieval in the era of language models

- Encode the **global information** in corpus; optimize in an **end-to-end way**
- The semantic-level **association** extending beyond mere signal-level matching

## Pros of generative retrieval

Information retrieval in the era of language models

- Encode the **global information** in corpus; optimize in an **end-to-end way**
- The semantic-level **association** extending beyond mere signal-level matching
- Constraint decoding over **thousand-level vocabulary**
- Internal index which **eliminates** large-scale external index

## Cons of generative retrieval: Scalability

- Highly dynamic corpora
  - Document addition, removal and updates
  - How to keep such GR models up-to-date?
  - How to learn on new data without forgetting old ones?

## Cons of generative retrieval: Scalability

- Highly dynamic corpora
  - Document addition, removal and updates
  - How to keep such GR models up-to-date?
  - How to learn on new data without forgetting old ones?
- Multi-modal/granularity/language search tasks
  - Different search tasks leverage very different indexes
  - How to unify different search tasks into a single generative form?
  - How to capture task specifications while obtaining the shared knowledge?

## Cons of generative retrieval: Controllability

For an issue, it is often unclear what modeling knobs one should turn to fix the model's behavior

## Cons of generative retrieval: Controllability

For an issue, it is often unclear what modeling knobs one should turn to fix the model's behavior

- Interpretability
  - Black-box neural models
  - How to provide credible explanation for the retrieval process and results?

## Cons of generative retrieval: Controllability

For an issue, it is often unclear what modeling knobs one should turn to fix the model's behavior

- Interpretability
  - Black-box neural models
  - How to provide credible explanation for the retrieval process and results?
- Debuggable
  - Attribution analysis: how to conduct causal traceability analysis on the causes, key links and other factors of specific search results?
  - Model editing: how to accurately and conveniently modify training data or tune hyperparameters in the loss function?

## Cons of generative retrieval: Controllability

For an issue, it is often unclear what modeling knobs one should turn to fix the model's behavior

- Interpretability
  - Black-box neural models
  - How to provide credible explanation for the retrieval process and results?
- Debuggable
  - Attribution analysis: how to conduct causal traceability analysis on the causes, key links and other factors of specific search results?
  - Model editing: how to accurately and conveniently modify training data or tune hyperparameters in the loss function?
- Robustness
  - When a new technique enters into the real-world application, it is critical to know not only how it works in average, but also how would it behave in abnormal situations

## Cons of generative retrieval: User-centered

Searching is a **socially** and **contextually** situated activity with diverse set of goals and needs for support that must not be boiled down to a combination of text matching and text generating algorithms [Shah and Bender, 2022]

## Cons of generative retrieval: User-centered

Searching is a **socially** and **contextually** situated activity with diverse set of goals and needs for support that must not be boiled down to a combination of text matching and text generating algorithms [Shah and Bender, 2022]

- Human information seeking behavior
- Transparency
- Provenance
- Accountability

## Cons of generative retrieval: Performance

The current performance of GR can only be compared to the **index-retrieval** stage of traditional methods, and it has **not yet** achieved the additional improvement provided by **re-ranking**

## So much to do ...

- **Closed-book:** The language model is the only source of knowledge leveraged during generation, e.g.,
  - Capturing document ids in the language models
  - Language models as retrieval agents via prompting
- **Open-book:** The language model can draw on external memory prior to, during and after generation, e.g.,
  - Retrieve-augmented generation of answers
  - Tool-augmented generation of answers

## So much to do ...

Cater for long-term effects

- How to combine the short-term relevance goal with long-term goals such as diversity

# So much to do ...

Cater for long-term effects

- How to combine the short-term relevance goal with long-term goals such as diversity

Address needs of interactive environments

- Interactive systems must operate under high degrees of uncertainty
  - User feedback, non-stationarity, exogenous factor, user preferences, ...

# So much to do ...

Cater for long-term effects

- How to combine the short-term relevance goal with long-term goals such as diversity

Address needs of interactive environments

- Interactive systems must operate under high degrees of uncertainty
  - User feedback, non-stationarity, exogenous factor, user preferences, ...

Searching/recommending slates of items

- Interface of many search/recommendation platforms requires showing combinations of results to users on the same page
- Different combinations may lead to different short vs. long-term outcomes
- Problem thus becomes combinatorial in nature, intractable for most applications

So much to do ...

Information retrieval meets large language models (LLMs)!

**Q & A**

**Thank you for joining us today!**

**All materials are available at**

**<https://sigir-ap2023-generative-ir.github.io/>**

## References

## References i

- M. Bevilacqua, G. Ottaviano, P. Lewis, W.-t. Yih, S. Riedel, and F. Petroni. Autoregressive search engines: Generating substrings as document identifiers. In *Advances in Neural Information Processing Systems*, pages 31668–31683, 2022.
- J. Chen, R. Zhang, J. Guo, Y. Fan, and X. Cheng. Gere: Generative evidence retrieval for fact verification. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2184–2189, 2022a.
- J. Chen, R. Zhang, J. Guo, Y. Liu, Y. Fan, and X. Cheng. Corpusbrain: Pre-train a generative retrieval model for knowledge-intensive language tasks. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 191–200, 2022b.
- J. Chen, R. Zhang, J. Guo, M. de Rijke, W. Chen, Y. Fan, and X. Cheng. Continual learning for generative retrieval over dynamic corpora. In *Proceedings of the 32nd ACM Conference on Information and Knowledge Management*, 2023a.
- J. Chen, R. Zhang, J. Guo, M. de Rijke, Y. Liu, Y. Fan, and X. Cheng. A unified generative retriever for knowledge-intensive language tasks via prompt learning. In *46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1448–1457, 2023b.

## References ii

- N. De Cao, G. Izacard, S. Riedel, and F. Petroni. Autoregressive entity retrieval. In *International Conference on Learning Representations*, 2021.
- R. Deffayet, T. Thonet, D. Hwang, V. Lehoux, J.-M. Renders, and M. de Rijke. Sardine: A simulator for automated recommendation in dynamic and interactive environments. *ACM Transactions on Recommender Systems*, Submitted.
- L. Heck and S. Heck. Zero-shot visual slot filling as question answering. *arXiv preprint arXiv:2011.12340*, 2020.
- J. D. M.-W. C. Kenton and L. K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.
- H. Lee, S. Yang, H. Oh, and M. Seo. Generative multi-hop retrieval. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1417–1436, 2022.
- Y. Li, N. Yang, L. Wang, F. Wei, and W. Li. Multiview identifiers enhanced generative retrieval. In *61st Annual Meeting of the Association for Computational Linguistics*, pages 6636–6648, 2023.
- J. Ma, T. Sun, and X. Zhang. Time highlighted multi-interest network for sequential recommendation. *Computers, Materials & Continua*, 76(3), 2023.

## References iii

- S. V. Mehta, J. Gupta, Y. Tay, M. Dehghani, V. Q. Tran, J. Rao, M. Najork, E. Strubell, and D. Metzler. Dsi++: Updating transformer memory with new documents. *arXiv preprint arXiv:2212.09744*, 2022.
- D. Metzler, Y. Tay, D. Bahri, and M. Najork. Rethinking search: Making domain experts out of dilettantes. *SIGIR Forum*, 55(1):1–27, 2021.
- T. Murayama. Dataset of fake news detection and fact verification: a survey. *arXiv preprint arXiv:2111.03299*, 2021.
- U. Nadeem, N. Ziems, and S. Wu. Codedsi: Differentiable code search. *arXiv preprint arXiv:2210.00328*, 2022.
- M. Najork. Generative information retrieval (slides), 2023. URL [https://docs.google.com/presentation/d/191AeVzPkh20Ly855tKDkz1uv-1pHV\\_9GxfntiTJPUG/](https://docs.google.com/presentation/d/191AeVzPkh20Ly855tKDkz1uv-1pHV_9GxfntiTJPUG/).
- R. Pradeep, K. Hui, J. Gupta, A. D. Lelkes, H. Zhuang, J. Lin, D. Metzler, and V. Q. Tran. How does generative retrieval scale to millions of passages? In *Gen-IR@SIGIR 2023: The First Workshop on Generative Information Retrieval*, 2023.

## References iv

- S. Rajput, N. Mehta, A. Singh, R. H. Keshavan, T. Vu, L. Heldt, L. Hong, Y. Tay, V. Q. Tran, J. Samost, et al. Recommender systems with generative retrieval. *arXiv preprint arXiv:2305.05065*, 2023.
- R. Ren, W. X. Zhao, J. Liu, H. Wu, J.-R. Wen, and H. Wang. Tome: A two-stage approach for model-based retrieval. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 6102–6114, 2023.
- C. Shah and E. M. Bender. Situating search. In *Proceedings of the 2022 Conference on Human Information Interaction and Retrieval*, pages 221–232, 2022.
- W. Sun, L. Yan, Z. Chen, S. Wang, H. Zhu, P. Ren, Z. Chen, D. Yin, M. de Rijke, and Z. Ren. Learning to tokenize for generative retrieval. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
- Y. Tang, R. Zhang, J. Guo, J. Chen, Z. Zhu, S. Wang, D. Yin, and X. Cheng. Semantic-enhanced differentiable search index inspired by learning strategies. In *29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023a.

## References v

- Y. Tang, R. Zhang, J. Guo, M. de Rijke, W. Chen, and X. Cheng. Listwise generative retrieval models via a sequential learning process. *arXiv*, 2023b. Major revision.
- Y. Tay, V. Q. Tran, M. Dehghani, J. Ni, D. Bahri, H. Mehta, Z. Qin, K. Hui, Z. Zhao, J. Gupta, T. Schuster, W. W. Cohen, and D. Metzler. Transformer memory as a differentiable search index. In *Advances in Neural Information Processing Systems*, volume 35, pages 21831–21843, 2022.
- Z. Wang, Y. Zhou, Y. Tu, and Z. Dou. Novo: Learnable and interpretable document identifiers for model-based ir. In *Proceedings of the 32nd ACM Conference on Information and Knowledge Management*, 2023.
- P. Zhang, Z. Liu, Y. Zhou, Z. Dou, and Z. Cao. Term-sets can be strong document identifiers for auto-regressive search engines. *arXiv preprint arXiv:2305.13859*, 2023.
- W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.
- Y. Zhou, J. Yao, Z. Dou, L. Wu, P. Zhang, and J.-R. Wen. Ultron: An ultimate retriever on corpus with a model-based indexer. *arXiv preprint arXiv:2208.09257*, 2022.

- Y. Zhou, Z. Dou, and J.-R. Wen. Enhancing generative retrieval with reinforcement learning from relevance feedback. In *EMNLP 2023: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.
- S. Zhuang, H. Ren, L. Shou, J. Pei, M. Gong, G. Zuccon, and D. Jiang. Bridging the gap between indexing and retrieval for differentiable search index with query generation. In *Gen-IR@SIGIR 2023: The First Workshop on Generative Information Retrieval*, 2023.