

Lecture 9. Pre-training Methods in Information Retrieval

Yixing Fan, Xiaohui Xie, Yinqiong Cai, Jia Chen, Xinyu Ma, Xiangsheng

Li, Ruqing Zhang and Jiafeng Guo (2022), “Pre-training Methods in Information Retrieval”, Foundations and Trends in Information Retrieval: Vol. 16, No. 3, pp 178–317.

DOI: 10.1561/15000000100.

Pre-training Methods in Information Retrieval

Suggested Citation: Yixing Fan, Xiaohui Xie, Yinqiong Cai, Jia Chen, Xinyu Ma, Xiangsheng Li, Ruqing Zhang and Jiafeng Guo (2022), "Pre-training Methods in Information Retrieval", Foundations and Trends® in Information Retrieval: Vol. 16, No. 3, pp 178–317. DOI: 10.1561/15000000100.

Yixing Fan
ICT, CAS
fanyixing@ict.ac.cn

Xiaohui Xie
Tsinghua University
xiexiaohui@mail.tsinghua.edu.cn

Yinqiong Cai
ICT, CAS
caiyinqiong18s@ict.ac.cn

Jia Chen
Tsinghua University
chenjia0831@gmail.com

Xinyu Ma
ICT, CAS
maxinyu17g@ict.ac.cn

Xiangsheng Li
Tsinghua University
lixsh6@gmail.com

Ruqing Zhang
ICT, CAS
zhangruqing@ict.ac.cn

Jiafeng Guo
ICT, CAS
guojiafeng@ict.ac.cn

Background

A Hierarchical View of IR

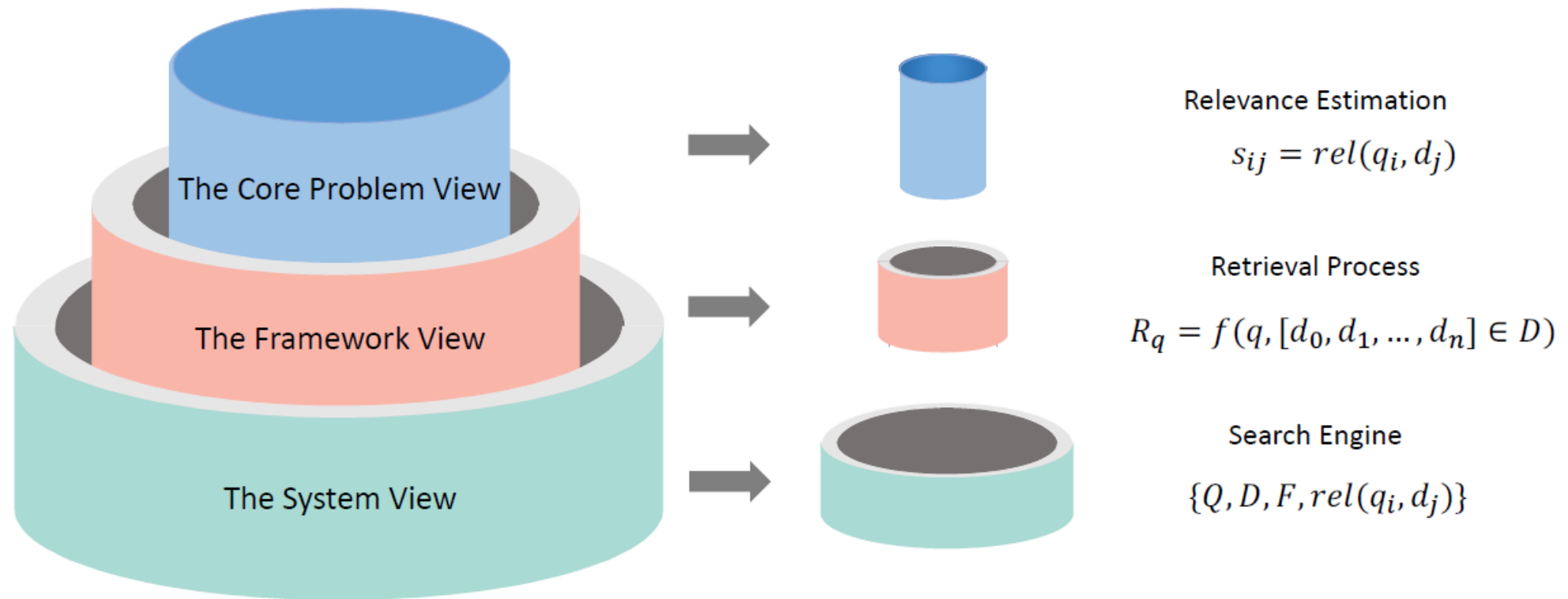


Figure 2.1: A Hierarchical View of IR

Core Problem View of IR

- Classical Retrieval Models
- Learning to Rank (LTR) Models
- Neural Retrieval Models

Framework View of IR

- Retrieval Architecture
 - Single-stage Retrieval ($n = 0$)
 - Two-stage Retrieval ($n = 1$)
 - Multi-stage Retrieval ($n \geq 2$)

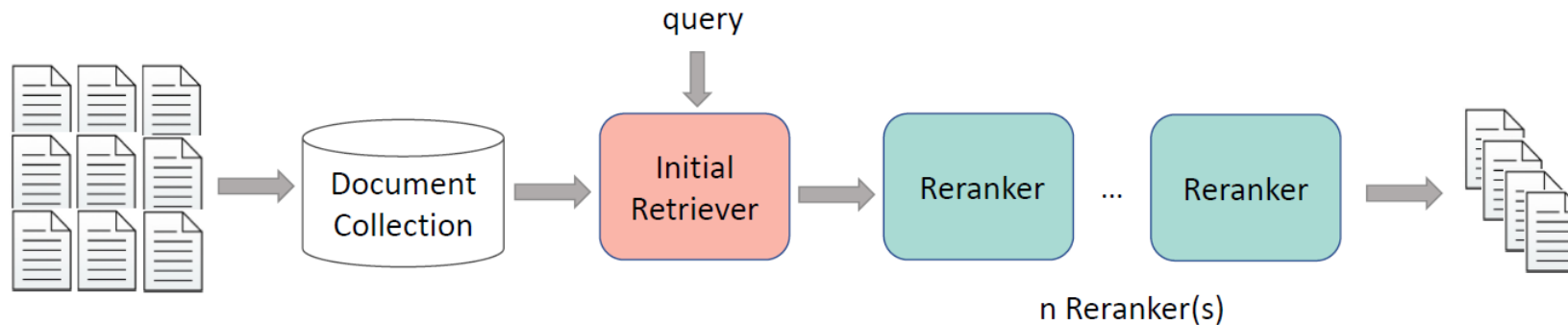


Figure 2.2: The retrieval architecture. According to the number of re-rankers, this retrieval process can be defined as Single-stage Retrieval ($n = 0$), Two-stage Retrieval ($n = 1$) and Multi-stage Retrieval ($n \geq 2$).

Retrieval vs. Re-ranker

- Traditional models such as BM25 are used to construct initial retrievers.
- Early stage re-rankers vs. later-stage re-rankers
 - Early-stage re-rankers will focus more on **efficiency**, but will pay more attention to **effectiveness** than retrievers
 - Later-stage re-rankers will focus more on **effectiveness**.

System View of IR

- Symbolic search system: focus more on “exact match”
- Neural search system: attempt to capture “semantic match”
 - Pro: easy to extend and scale
 - Con: less explainability and need for lots of data for training

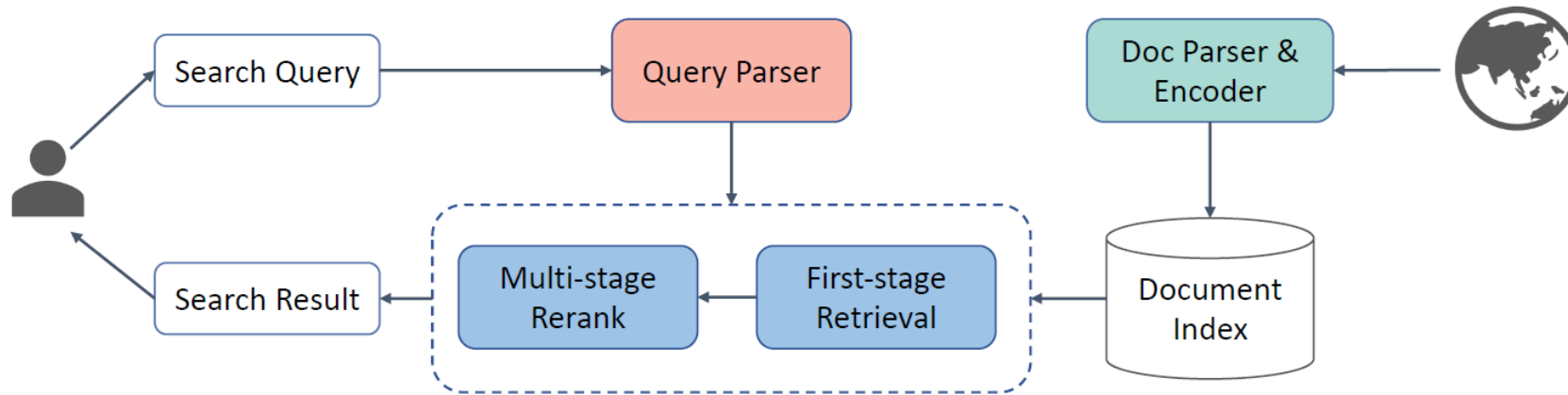


Figure 2.3: The framework of a practical search system.

Overview of PTMs in IR

- 1) PTMs are first applied to learn either good representations of texts or better interaction between text-pairs based on unlabeled datasets
- 2) The learned representations/interactions are then fine-tuning and used for downstream tasks
 - 1) **Full fine-tuning**: fine-tuning all weights with the data from the downstream task
 - 2) **Partial fine-tuning**: fine-tuning partial weights that are specific to the downstream task while freezing the other weights
 - 3) **Freezing the weights**: using the representation from the frozen weight to solve the downstream task

The development of PTMs in IR

- The first phase (the 2010s)
 - Word embedding methods have been investigated to develop meaningful representations of words
- The second phase
 - Transformer-based methods are proposed to gain better representations or interactions of texts by considering more sophisticated model structures and pre-training objectives

Word Embedding Methods

- Categorization
 - Word2vec
 - GloVe
 - Paragraph2vec
- Functions
 - Used to refine **term weighting schemes** in the inverted index (replace term frequency)
 - Applied to better estimate the **matching levels** of queries and documents.
 - Adopted to benefit crucial **IR-related tasks**, e.g., query suggestion and document summarization.

Transformer-based Methods (1/2)

- Word embedding methods cannot deal with the **context-dependent nature** of words and the issue of **polysemous**.
- Transformer-based methods are used to **estimate the relevance level** between the query and the document.
 - DPR (**representation-focused**): learns **dense embeddings for the document** with a BERT-based encoder, and **queries are encoded** with another independent BERT-based encoder. The outputs of the two encoders are then **fed into a “similarity” function** to obtain the relevance score.
 - MonoBERT (**interaction-focused**) takes the **concatenation of the query and document as the input** and **feeds the [CLS] vector output by BERT** to a feed-forward network to obtain the relevance score of the given query and document.

Transformer-based Methods (2/2)

- Transformer-based methods also **considers the trade-off between efficiency and effectiveness** according to the stages (**retrieval or reranking**) they targets.
 - PTMs are used to improve the performance of retrieval models (**sparse, dense or hybrid**).
- Different transformer-based methods are tailored for different components, i.e., “**Query parser**”, “**Doc Parser & Encoder**”, and “**Retrieval and Rerank**” in the search system.

Recent PTMs in IR

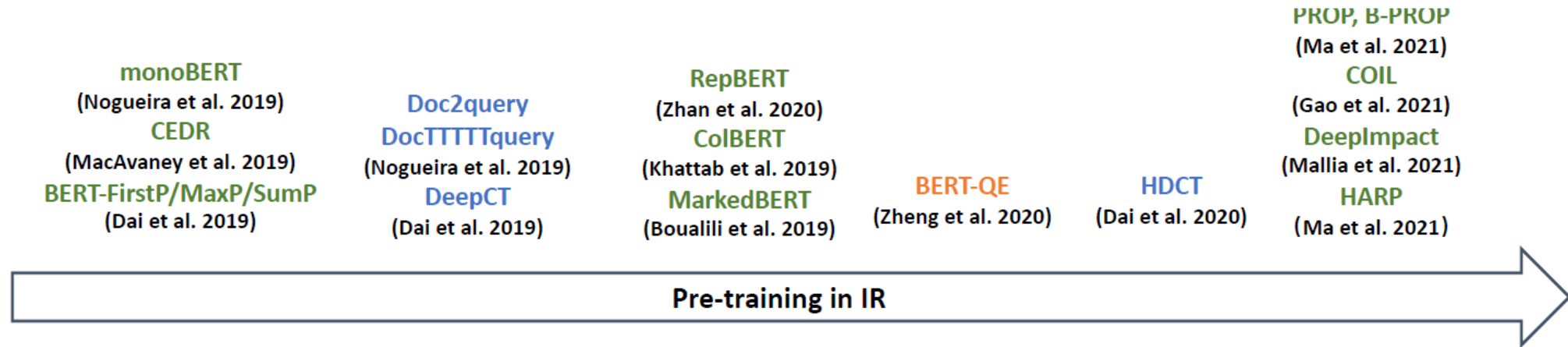


Figure 2.4: Recent PTMs in IR. “Orange”, “Green” and “Blue” refer to the “Query Parser”, “Retrieval and Rerank”, and “Doc Parser & Encoder” stages for which PTMs target respectively.

Pre-training Methods (PTM) Applied in the Retrieval Component

Basic Model Structure

- 1) Sparse Retrieval Models
 - Improve retrieval by obtaining semantic augmented sparse representations and index them with the inverted index for efficient retrieval.
- 2) Dense Retrieval Models
 - Project input texts (i.e., queries and documents) into standalone dense representations and turn to approximate nearest neighbor search algorithms for fast retrieval.
- 3) Hybrid Retrieval Models
 - Build sparse and dense retrieval models concurrently to absorb merits of both for better retrieval performance.

Sparse Retrieval Models

- Queries and documents are represented with high-dimensional sparse embeddings so that the inverted index can be still used for efficient retrieval
- Improving retrieval performance by
 - 1) enhancing the bag-of-words (BoW) representations in classical term-based methods
 - 2) mapping input texts into the “latent word” space
- Apply PTMs in sparse retrieval models
 - term re-weighting
 - document expansion
 - expansion + re-weighting
 - sparse representation learning

Four Architectures of Sparse Retrieval Models

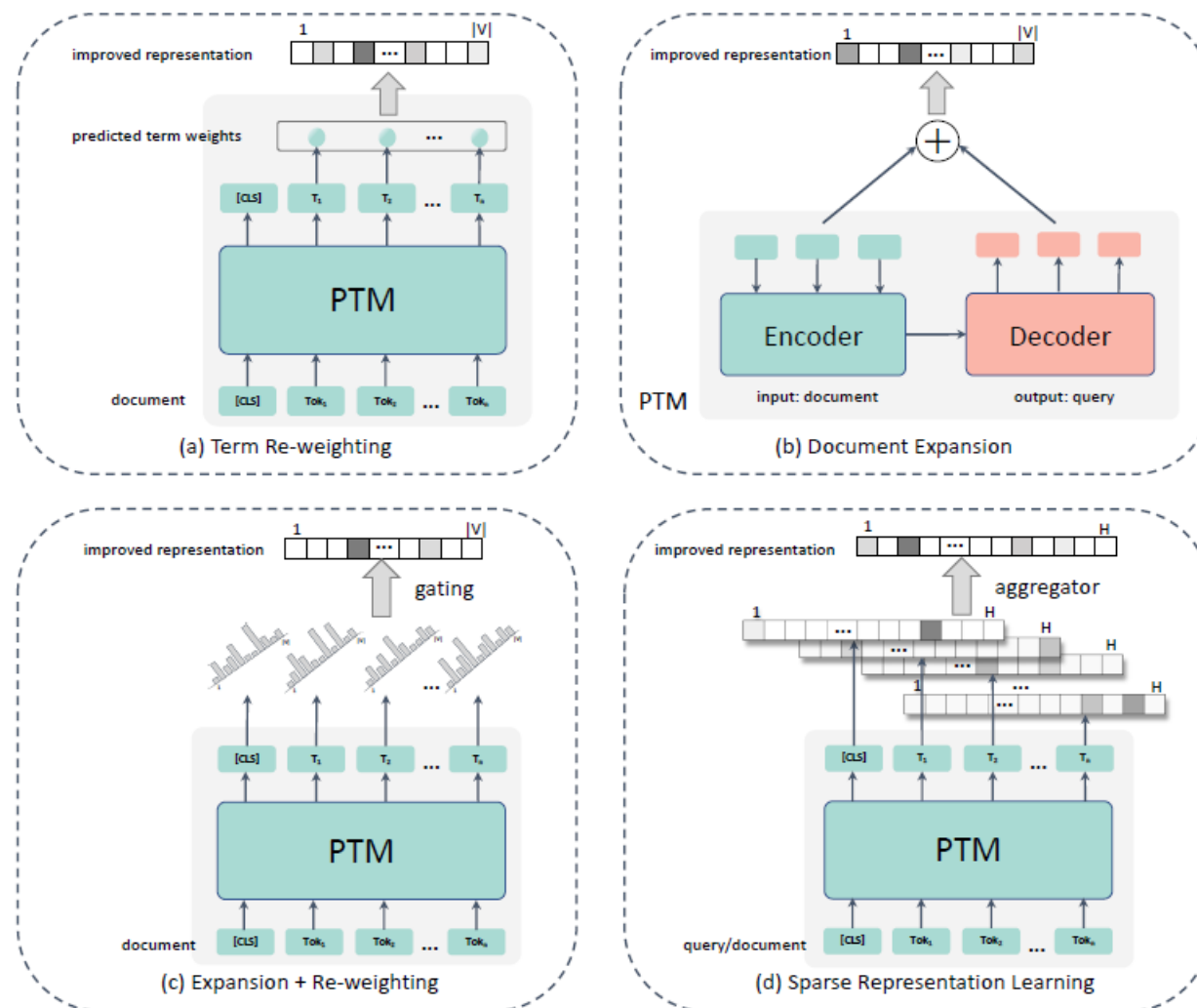
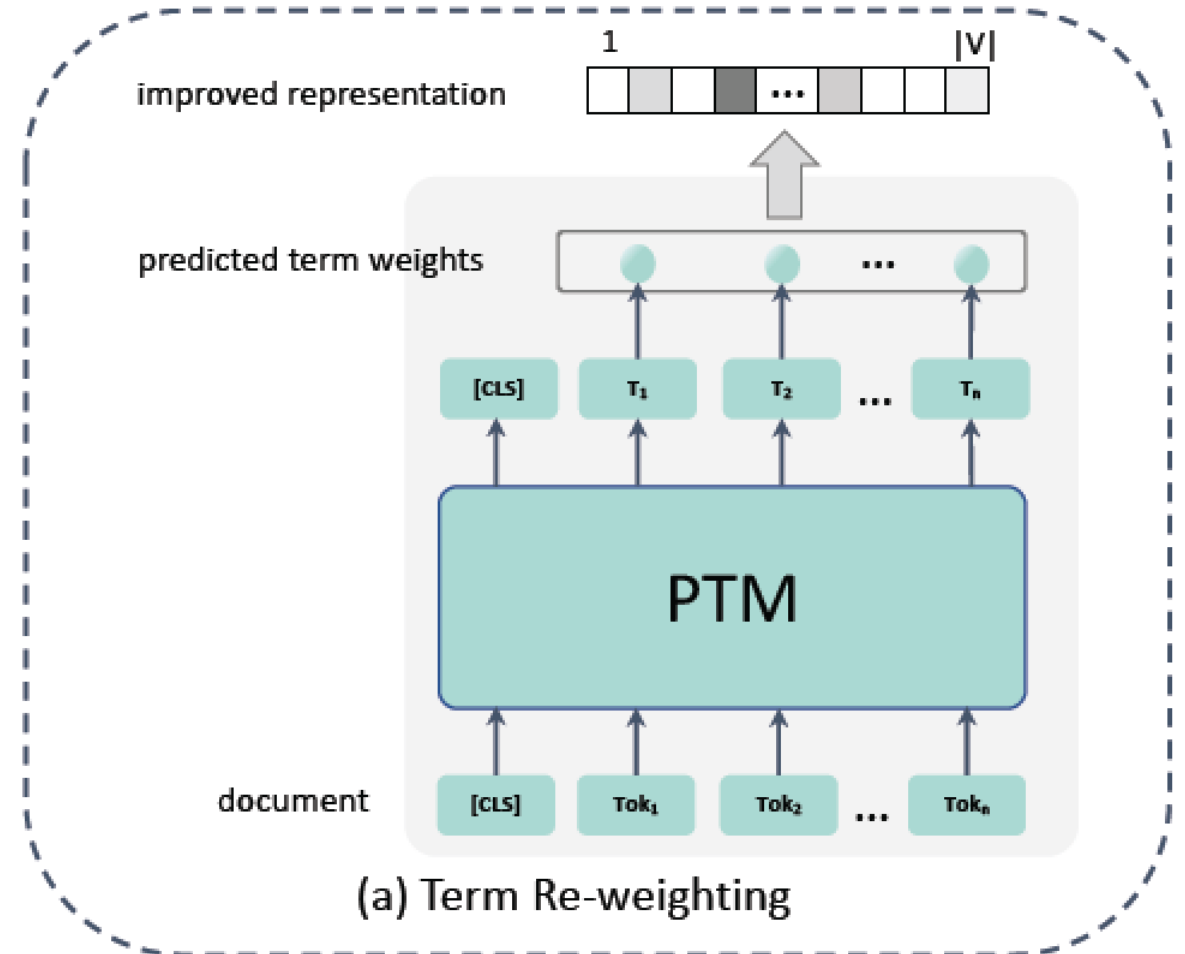


Figure 3.1: Four architectures of sparse retrieval models.

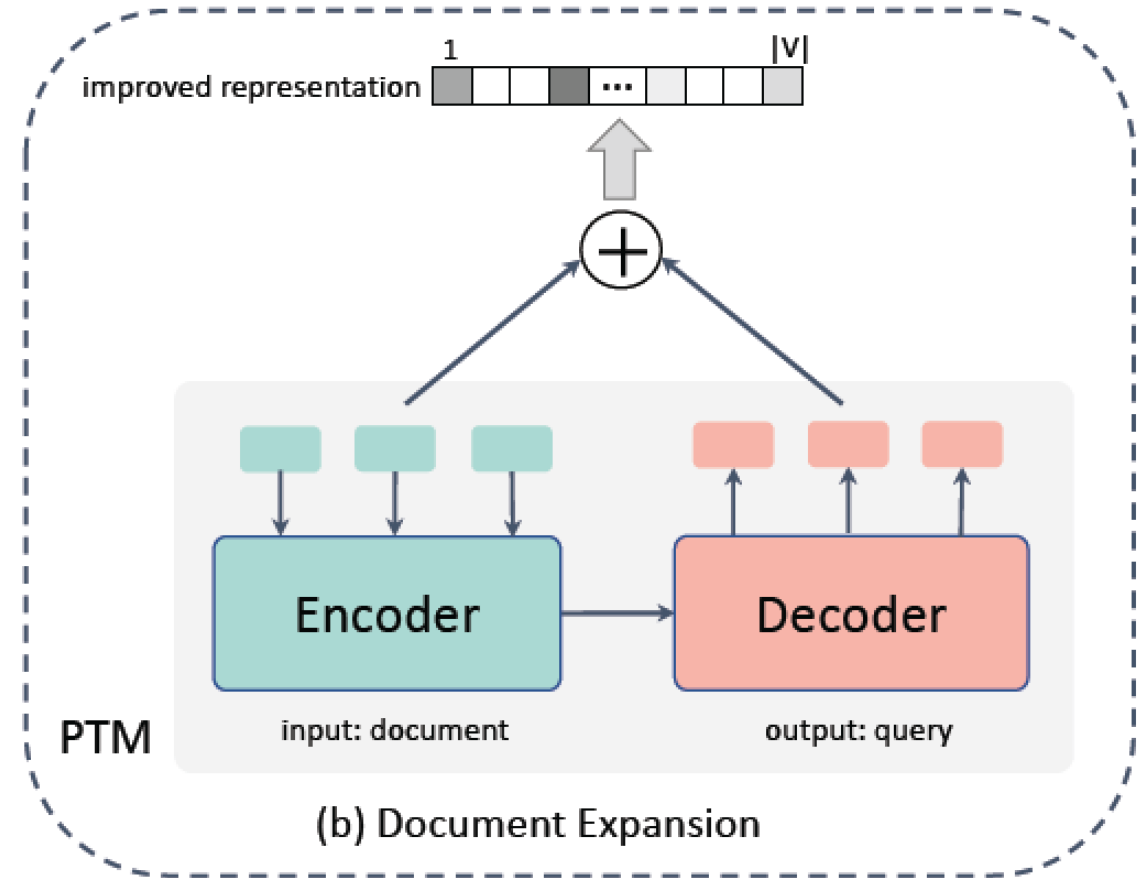
Term Re-weighting

- Measure term weights with contextual semantics, instead of term frequency (TF)
 - Leverage term weights estimated by pre-trained word embeddings to replace TF in the inverted index to improve the retrieval effectiveness
 - Utilized FastText to estimate the IDF field in the inverted index



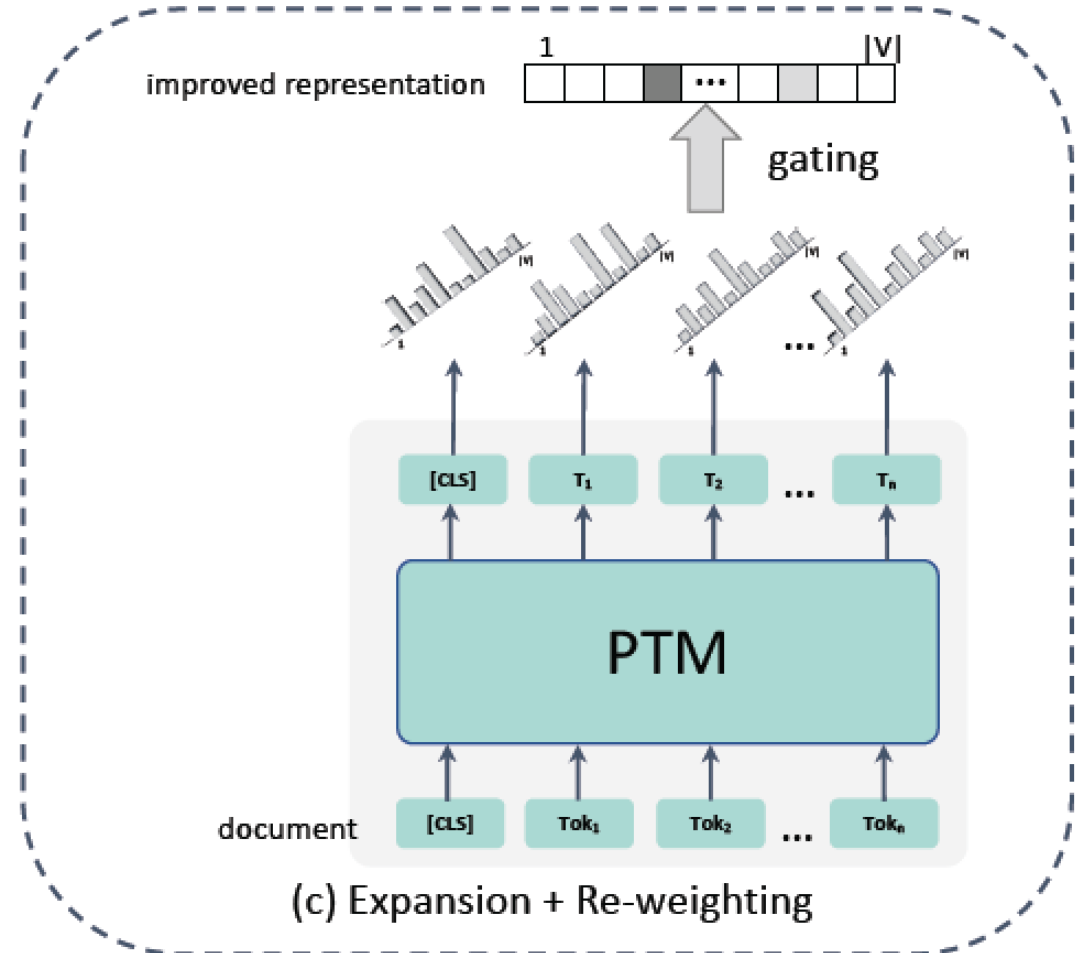
Document Expansion

- Augmenting the document with semantically related terms
 - Firstly **fine-tuned a pre-trained language model T5** with relevant query-document pairs.
 - The learned model **generates multiple queries for each document** and appends them to the original document.
 - Then, they used BM25 to retrieve relevant documents **based on the expanded document collection**.



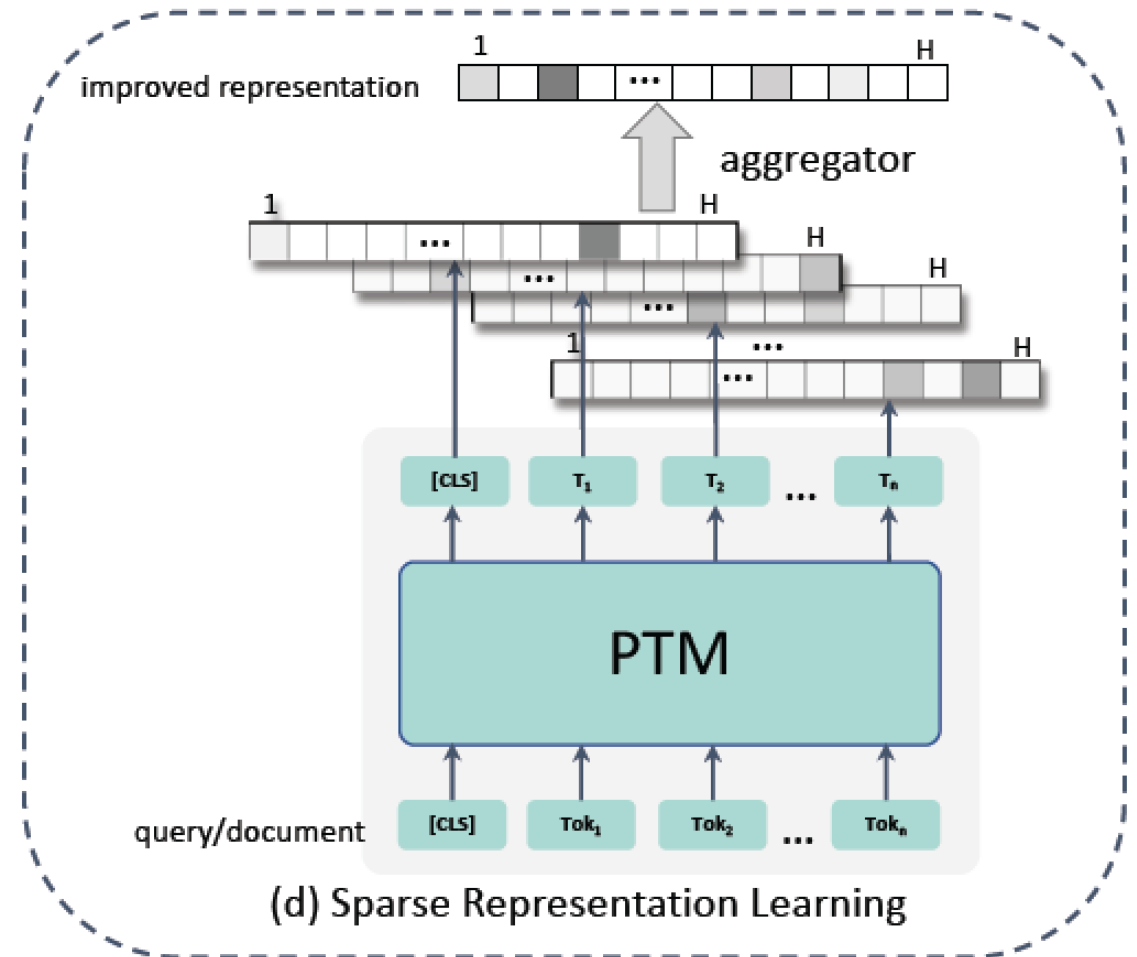
Expansion + Re-weighting

- Combine the idea of term re-weighting and document expansion
 - Generates multiple queries for each document and appends them to the original document.
 - Predict the term importance distribution in the vocabulary space based on contextual token embeddings got by BERT
 - Re-weight existing and expand terms simultaneously



Sparse Representation Learning

- The above methods improve document representations in explicit symbolic space
- Learn sparse embeddings for queries and documents in the latent word space



Dense Retrieval Models

- Turn to dense representations from sparse representations
 - Dense retrieval models **employ the dual-encoder architecture**, also known as Siamese network, to learn low-dimensional dense embeddings for queries and documents.
 - The learned dense representations **are indexed via approximate nearest neighbor (ANN)** search algorithms to support online search.
- A simple matching function (e.g., dot product or cosine similarity) is used to calculate the relevance scores based on the learned representations.

$$rel(q, d) = f(\phi_{PTM}(q), \varphi_{PTM}(d)),$$

where ϕ_{PTM} and φ_{PTM} are query and document encoders based on pre-training methods, and f is the similarity function.

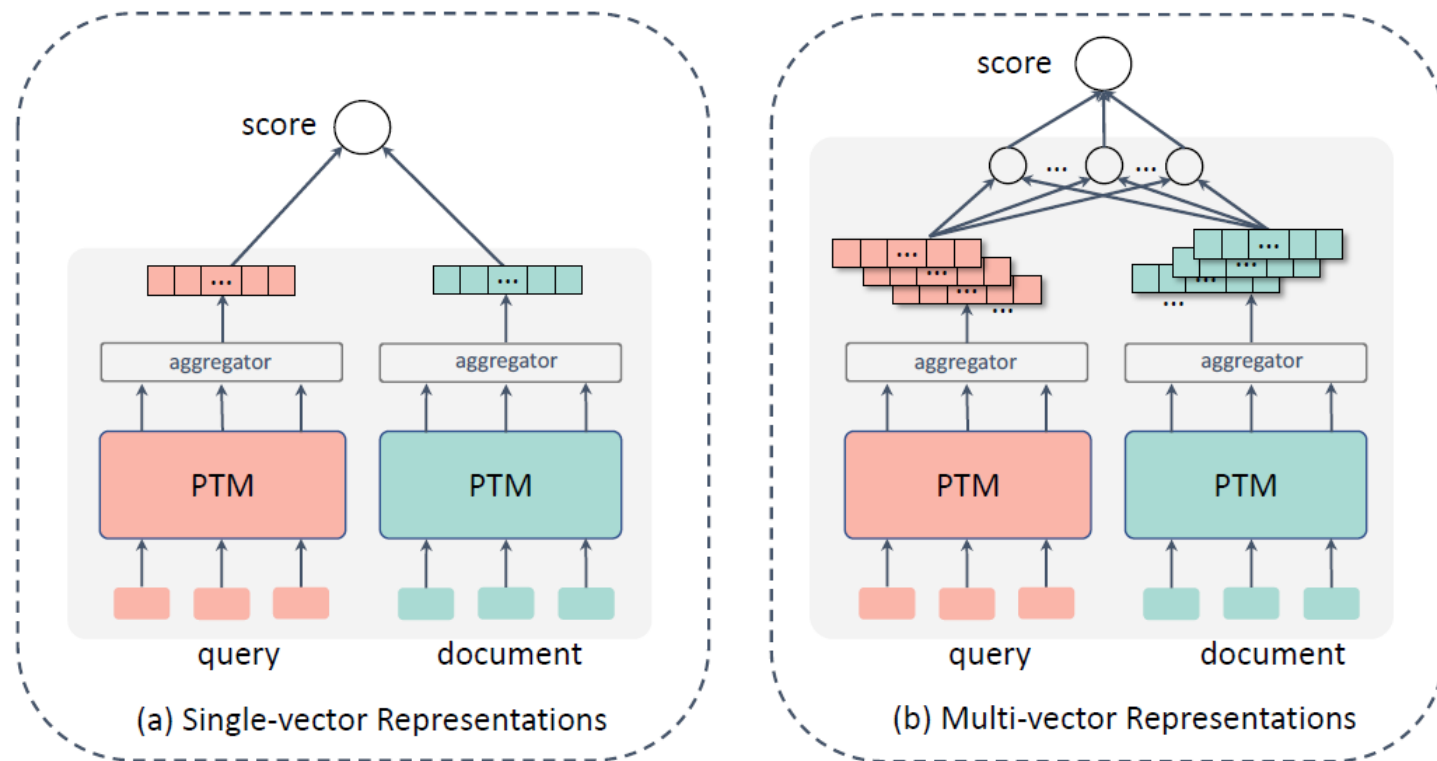


Figure 3.2: Basic architectures of dense retrieval models.

two dense retrieval families have emerged: **single-vector representations** (Figure 3.2 (a)), where the entire input text is represented by a single embedding, and **multi-vector representations** (Figure 3.2 (b)), where the input text is represented by multiple contextual embeddings.

Single-vector Representation

- Initially, some works used simple heuristic functions to **aggregate pre-trained word embeddings** and obtained dense representations for queries and documents.
- Except for obtaining dense query/document representations based on pre-trained embeddings, existing attempts at improving the quality of dense retrieval models focuses on **finding more powerful representation learning functions**.
- DPR
 - Learn dense embeddings for queries and passages with **two independent BERT-based encoder**
 - Relevance scores are calculated with **the inner product operation** between query and document representations

Multi-vector Representation (1/2)

- Obtain multiple vectors for queries and documents
 - Based on the learned pre-trained word2vec embedding model, **query words are mapped into the input space** and **document words are mapped into the output space**.
 - The final relevance score is calculated with **aggregated cosine similarities between all query-document word pairs**.
- Except for the pre-trained word embeddings, there are also a number of works that **employ pre-trained models to learn query/document representations** for IR.
- ColBERT
 - Generate contextualized term embeddings for queries and documents with a BERT-based dual-encoder
 - Employ the MaxSim operator to obtain the matching score.

Multi-vector Representation (2/2)

- ME-BERT
 - Takes the contextualized embedding of CLS as the single-vector query representation and the first m contextualized token embeddings as the multi-vector document representation.
 - The largest inner product between each document vector with the query vector is taken as the relevance score.

Hybrid Retrieval Models

- Sparse retrieval models take a (latent) word as the unit of representations and calculate the matching score based on exact matching signals.
- Dense retrieval methods learn dense embeddings for queries and documents and the relevance is evaluated with soft matching signals.

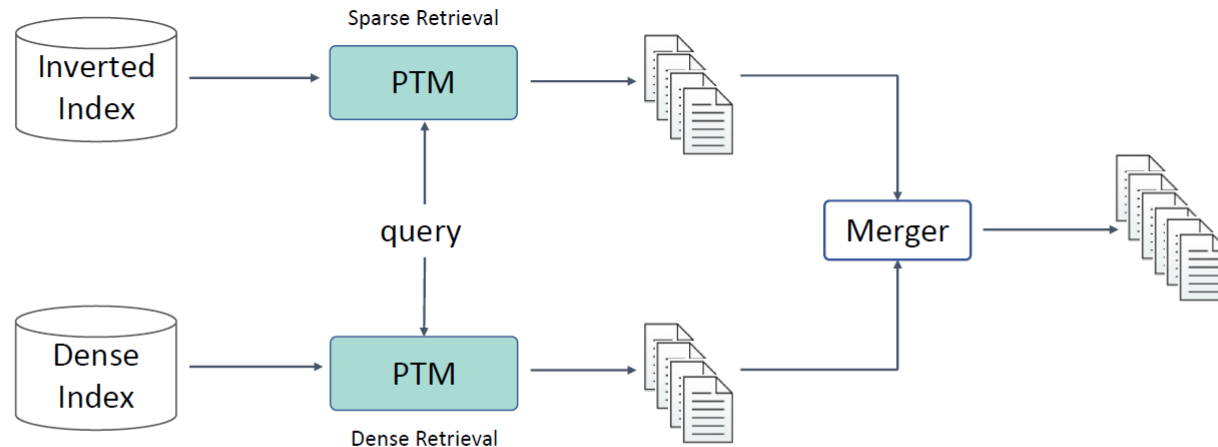


Figure 3.3: The architecture of hybrid retrieval models.

Methods

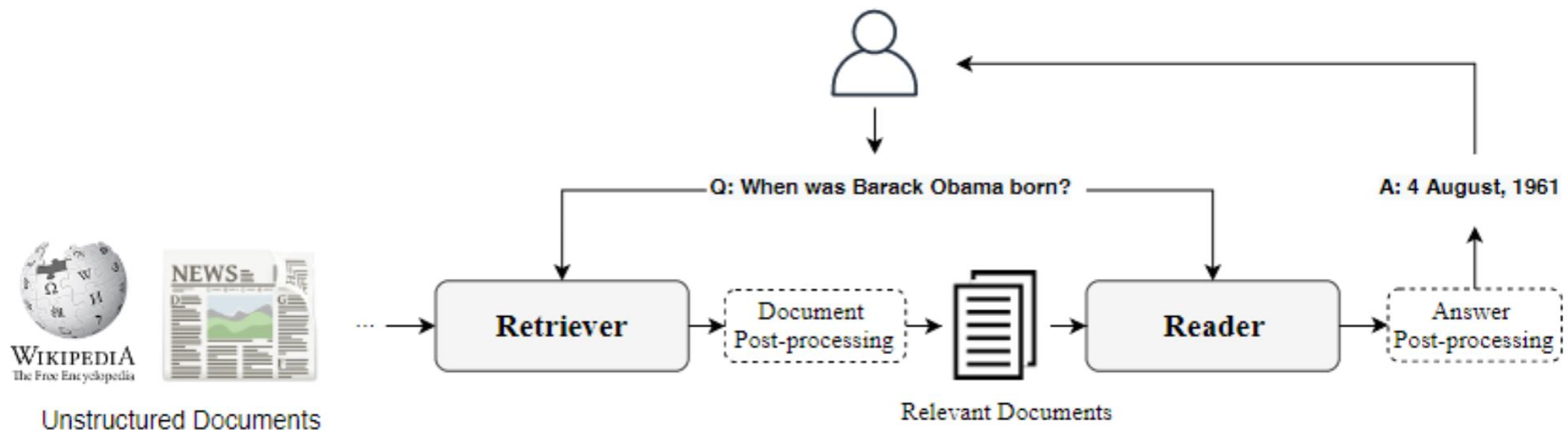
- A number of works were proposed to combine pre-trained word embeddings with term-based models for the retrieval component.
- Only relying on pre-trained word embeddings to build the retrieval model always shows poor performance, unless combining it with the term-based retrieval method.
- A more simple and direct way to build a hybrid retrieval model is to linearly combine matching scores of a sparse retrieval system and a dense retrieval system using a single trainable weight.

Joint Learning with Other Components (1/2)

- For different applications, the retrieval component can be learned with downstream components end-to-end, e.g., re-rankers for ad-hoc retrieval and readers for OpenQA.
- Joint Learning with Index
 - To support rapid online search, retrieval systems usually build an index for all documents in the collection.
 - The dense retrieval methods usually rely on ANN search algorithms to perform efficient retrieval.
- Joint Learning with Re-ranker
 - A joint training method for dense retrieval and re-ranking, where the relevance information can be transferred between the two components with a unified list-wise training approach.

Joint Learning with Other Components (2/2)

- Joint Learning with Reader
 - RAG combines a pre-trained dual-encoder (DPR) as the retriever with a pre-trained sequence-to-sequence model (BART) as the generator for OpenQA tasks.
 - The query encoder and the generator are fine-tuned end-to-end with the fixed document encoder.



Pre-training Methods Applied in the Re-ranking Component

Multi-stage Cascaded Architecture

- After the efficient first-stage retriever, there can be a stack of complex re-rankers in the re-ranking stage where the input of each re-ranker comes from the previous one.
- PTMs are often employed to re-rank a small set of candidates provided from the first-stage retriever.

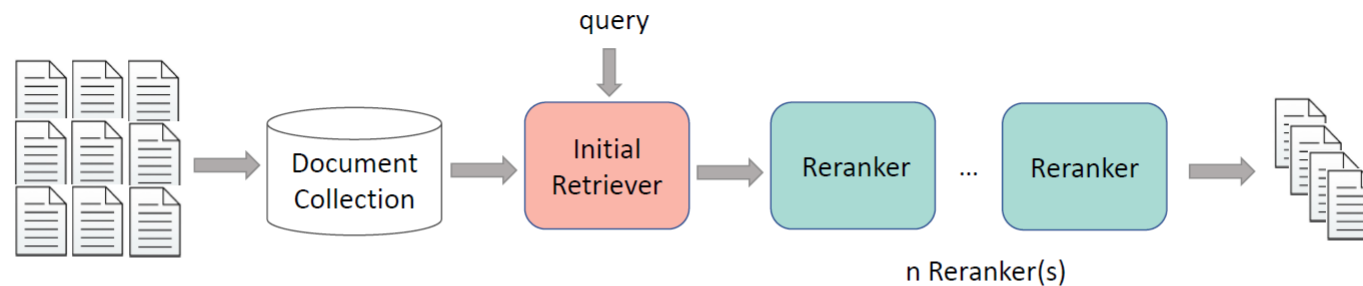
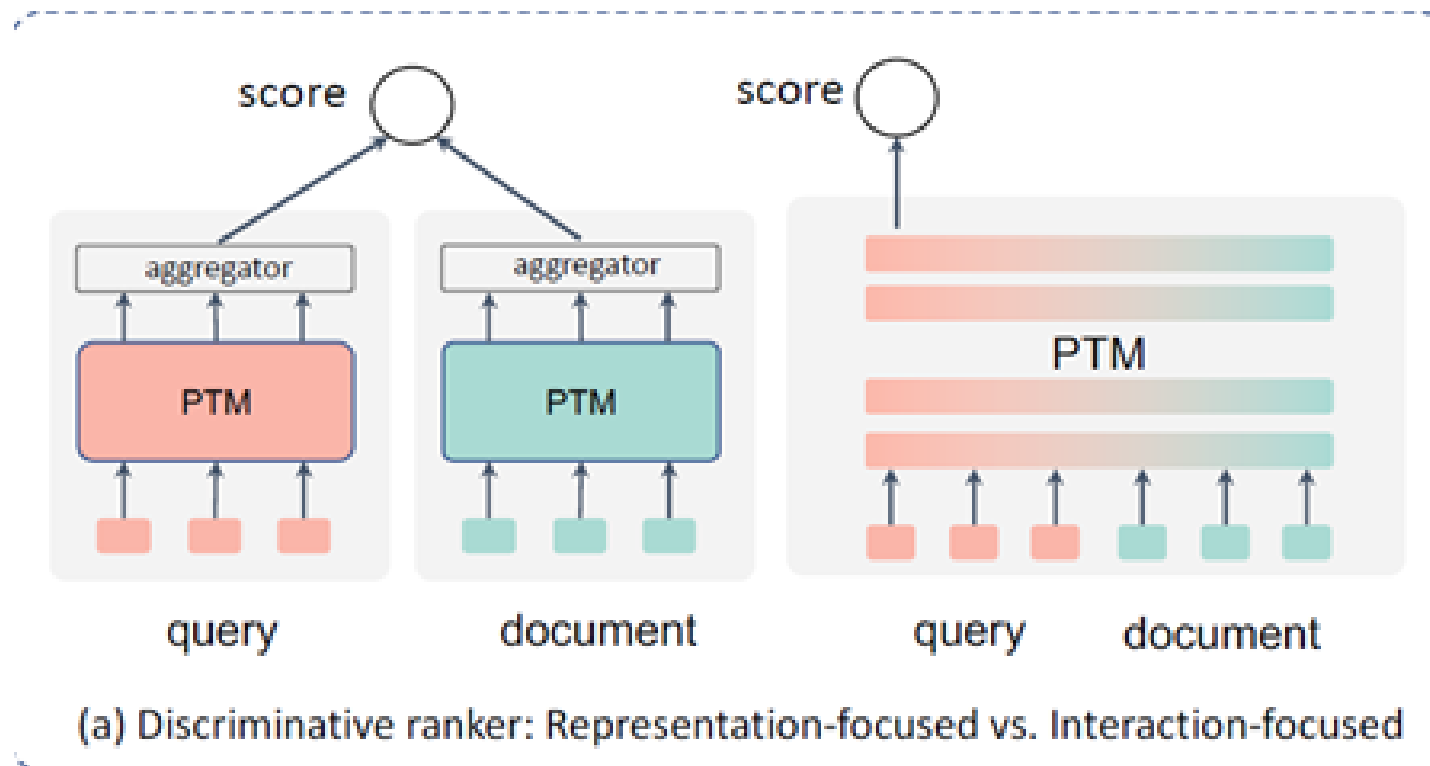


Figure 2.2: The retrieval architecture. According to the number of re-rankers, this retrieval process can be defined as Single-stage Retrieval ($n = 0$), Two-stage Retrieval ($n = 1$) and Multi-stage Retrieval ($n \geq 2$).

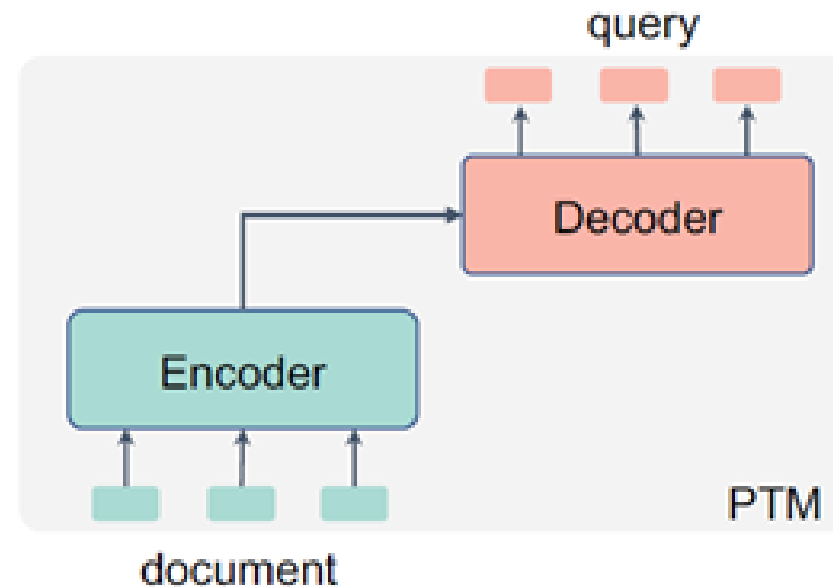
Basic Model Architecture (1/3)

- **Discriminative Ranking Models:** model $P(r, d|q)$ by directly learning a relevance “classifier” from labeled data



Basic Model Architecture (2/3)

- **Generative Ranking Models:** approximate the true relevance distribution $P(r|q, d)$ by modeling the generative process between queries and documents



(b) Generative ranker

Basic Model Architecture (3/3)

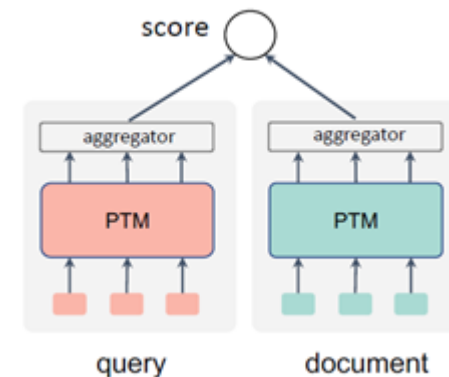
- **Hybrid Retrieval Models:** joint learn the discriminative model and generative model to leverage merits of both for better ranking performance.

Discriminative Ranking Models

- From 2015-2018, applying PTMs in the re-ranking component focused on **leveraging the pre-trained word embedding** such as word2vec and GloVe into **discriminative ranking models**.
 - These word embeddings are mainly used to **initialize the embedding layer of ranking models**, and **other components** are **usually learned from scratch**.
- Start with BERT, both pre-trained word representations and interactions can be “transferred” to the ranking model.
 - The “pre-train and fine-tune” paradigm: fine-tunes the whole pre-trained model and **only a lightweight task specific classification layer is learned from scratch**.

Representation-focused Models (1/3)

- Adopt a bi-encoder architecture and encode queries and documents separately, and then the relevance score is computed with simple similarity functions between representations of queries and documents.
 - ϕ_{PTM} and ϕ_{PTM} are PTMs which take the raw text of the query or the document as the input, and output one dense representation for each, respectively.
 - ϕ_{PTM} and ϕ_{PTM} could share the parameters or not.
 - Then, the relevance is computed by simple similarity functions f like cosine or MLP.



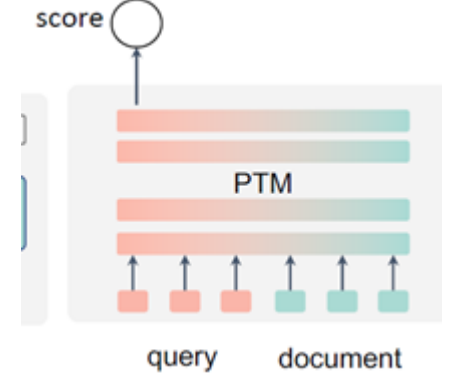
Representation-focused Models (2/3)

- In the early days, representation-focused methods often **employ pretrained word embeddings to initialize the representation of input tokens**, and the **remaining parameters are all randomly initialized**.
 - Train 50-dimensional word embeddings on Wikipedia and Weibo data using the Word2Vec.
 - Feed the word embeddings into convolutional neural networks to obtain text sequence representations.
 - Compute the relevance score using a MLP based on the two text sequence representations.
- More recently, **the Transformer-based PTMs are introduced to finetune the entire model on downstream tasks**, rather than just initializing the word embedding layer.
 - Utilizing the BERT to encode the query and the document separately, and take the [CLS] embedding of the last layer as their representations and then calculate the ranking score via cosine similarity.

Representation-focused Models (3/3)

- In general, discriminative ranking models can be fine-tuned using the pointwise, pairwise, or listwise learning objectives following the learning to rank literature.
- The Transformer-based PTMs usually limit their input length to 512 due to the quadratic time and memory complexity of self-attention.
- Long documents that contained more than 512 tokens will be truncated before being fed into the model, and more techniques about handling long documents for Transformer-based PTMs will be introduced.

Interaction-focused Models



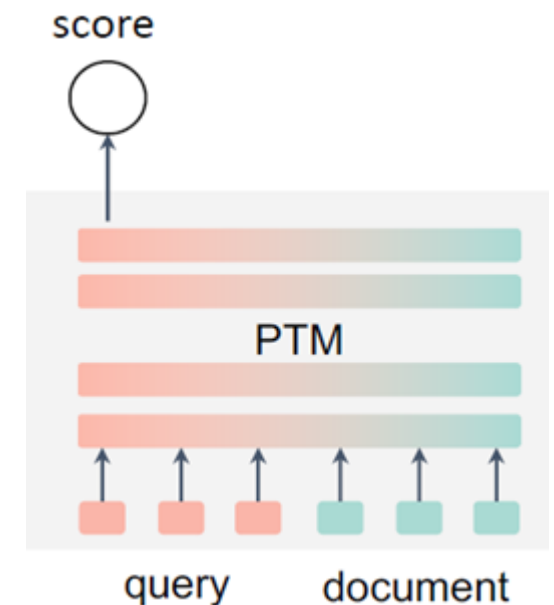
- Capture low-level interactions between terms in query document pairs, and then calculate the relevance score based on their interaction features.
- The interaction-focused method could be abstracted as:

$$rel(q, d) = f(\eta_{PTM}(q, d))$$

- where η_{PTM} is the interaction function based on PTMs, and f is the scoring function that estimates the relevance score according to the interaction features.
- The input for η_{PTM} is a concatenation of the query and the document.
- The interaction of the query and the document could be modeled inside the η_{PTM} with the self-attention mechanism.

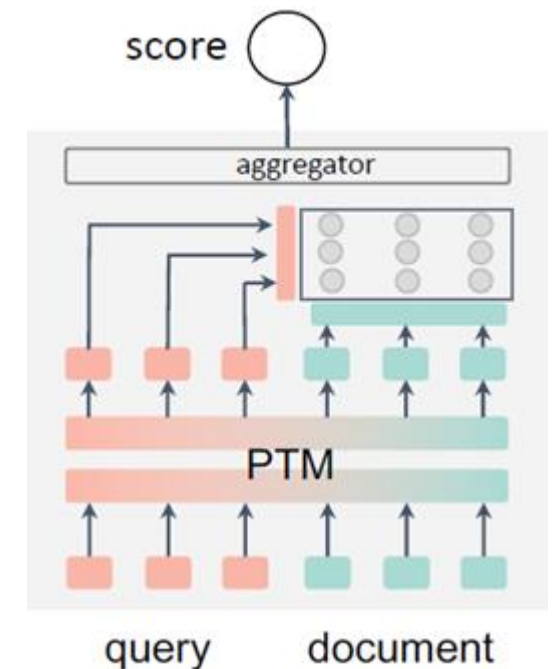
Three Typical Interaction-Focused Discriminative Ranking Models (1/3)

- Take the concatenation of the query and the passage as inputs of the BERT.
- Feed the [CLS] vector to a feed-forward network to obtain the relevance score.
- Take the pointwise loss function, i.e., the cross-entropy loss, to fine-tune the BERT model on the MS MARCO passage ranking task.



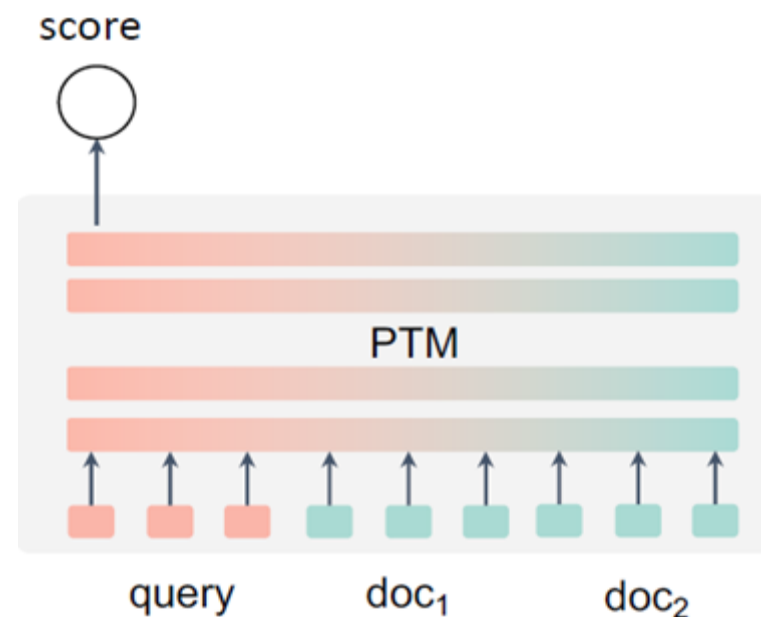
Three Typical Interaction-Focused Discriminative Ranking Models (2/3)

- Leverage the contextualized word embedding of BERT to build a similarity matrix.
- Feed into an existing interaction-focused model such as DRMM or KNRM.
- The [CLS] vector is also incorporated in CEDR to enhance the model's signals.
- CEDR is trained using pairwise hinge loss.



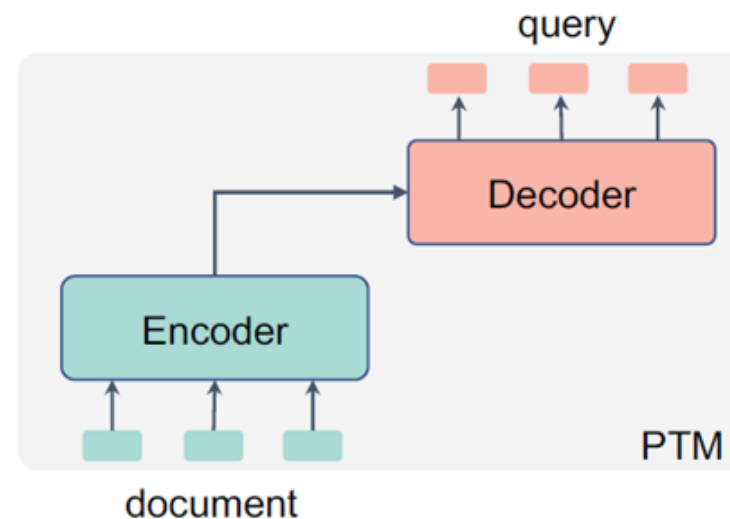
Three Typical Interaction-Focused Discriminative Ranking Models (3/3)

- Take a query and two passages as input, and concatenates them as a sequence to predict the correct relative order between these two passages.
- Explicitly model the document comparison for pairwise learning objectives.
- Due to the length limitation of the BERT, the whole sequence is truncated to 512 tokens and each passage can have at most 223 tokens.



Generative Ranking Models

- Approximate the true relevance distribution between queries and documents.
- Statistical language models like the **query likelihood model** consider the query generation process which ranks documents according to how likely query terms are generated from a document.
- For word embedding-based PTMs, they can be easily incorporated into statistical generative retrieval models to compute the semantic similarity between terms.



Applying Generative PTMs to Re-ranking

- 1) Query Generation process, which is inspired by the query likelihood model.
- 2) Relevance Generation process, which generates a specified relevance token given the query and the document.

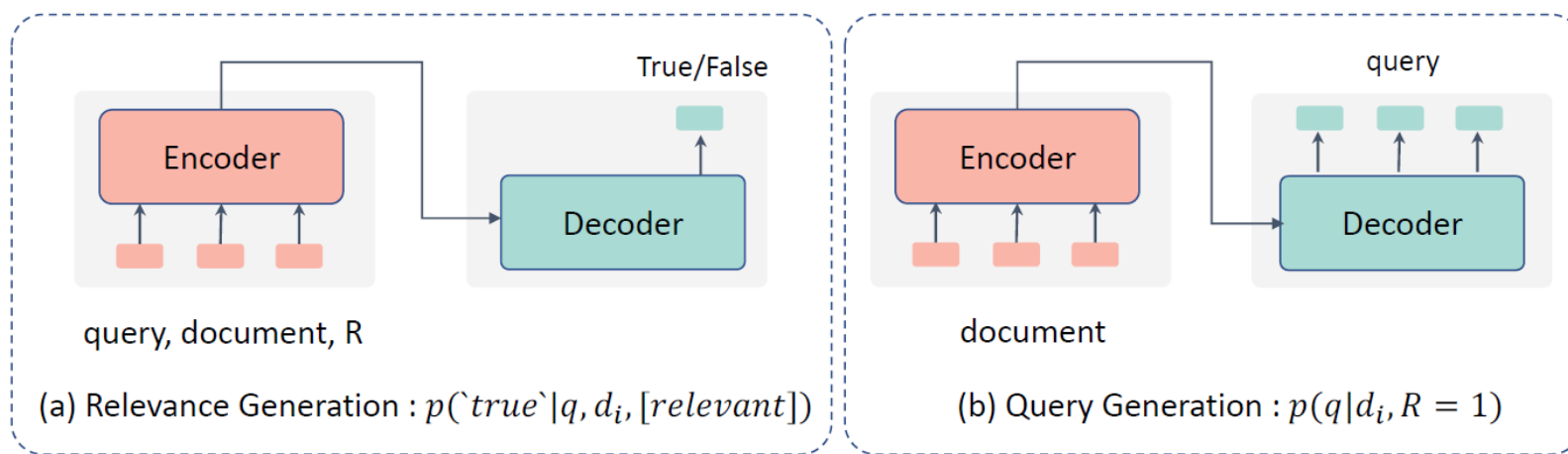
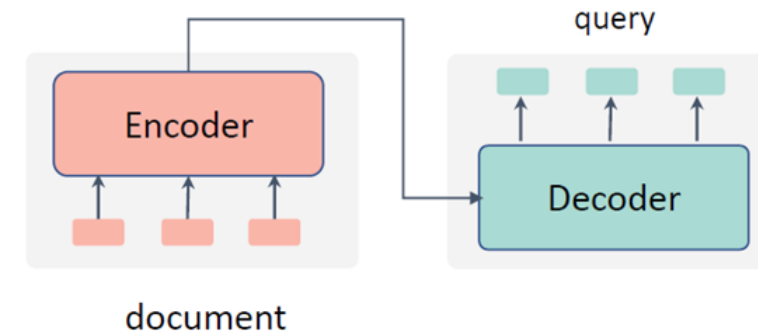


Figure 4.3: Two categories of generative ranking models.

Query Generation

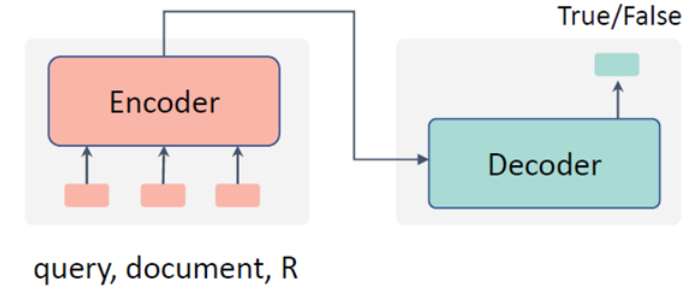


- Rank documents by the likelihood of generating the query from documents using generative PTMs like GPT and BART.
- The query generative models could be abstracted as

$$rel(q, d) = f(\phi_{PTM}(q|d)) = \prod_{i=1}^{|q|} \phi_{PTM}(q_i|d),$$

- where ϕ_{PTM} is the **generative PTMs** and f is a multiplication function \prod .
- Given the document d , each query term q_i is generated one by one.
- The relevance score is thus obtained by multiplying their normalized probabilities.
- Take generative PTMs like GPT and BART to estimate the probability in generating queries.

Relevance Generation



- Generating specified relevance tokens by feeding the concatenation of the document and the query into the generative PTMs.
- The probabilities of these relevance labels are treated as relevance scores.
- The relevance generative models could be reformulated as:

$$rel(q, d) = f(\phi_{PTM}(t|q, d))$$

- where t is the relevance tokens.
- The relevance generation is a classification task as the model is trained using pointwise loss function on relevance tokens and ranks documents by the probability of predicting the target relevance token.

Using the Generative PTMs T5 for Modeling Relevance Generation

- Devise a text-to-text template for the ranking task where the input is “Query: [q] Document: [d] Relevant:” and the output is “true” or “false”.
- T5 is fine-tuned to generate the target tokens instead of directly producing relevance probabilities.
- The probability of the “true” token is used to represent the document relevance score, which is normalized with softmax function over the logits of “true” and “false” tokens .
- Other target tokens like “yes/no” perform worse than the “true/false” tokens.

Similar to Prompt Learning

- The model is guided to predict the “label” based on prompts.
 - A template and a verbalizer are needed to design first for a given task, where the template is used to transform the original text to a specific form.
 - The verbalizer is used to project original labels to some words which are fit for the template.
- Sentiment classification task as an example
 - The template is “[text] It is [mask]” in which the token [text] represents the original text, and the token [mask] stands for the verbalized words such as “great” and “terrible”.
 - These two words are mapped from the positive label and the negative label, respectively.
 - The PTMs are trained to predict the probability distribution on the [mask] position given the text with a specific form.
- How to leverage the prompt learning to improve the few-shot learning in IR has not been explored at this point.

Hybrid Models

- Combining the generative and the discriminative modeling leads to the hybrid models.
- A multi-task learning approach jointly learn the discriminative and the generative relevance modeling in a unified pre-trained model.
- Leverage the generative PTMs (i.e., BART) or the discriminative PTMs (i.e., BERT) to learn discriminative ranking tasks as well as other language generation tasks, such as the query generation task, questions generation task, and anchor text generation task.
 - Feed the document and the query into the encoder and the decoder respectively.
 - Generate the query in a sequence-to-sequence manner and calculate the relevance score by the last token of the entire sequence using a feedforward layer.

Advanced Topics

- The document length varies significantly across different domains, where PTMs often fail to address the long document due to the length restriction of the input.
- PTMs often consist of a large number of parameters which would increase the search latency.
- Two issues
 - Long Document Processing Techniques
 - Passage Score Aggregation

Long Document Processing Techniques

- Due to the quadratic time and memory complexity of self-attention mechanism in modern Transformer-based PTMs, the length limit of input is always up to 512.
- Segment the long document text into smaller chunks that can be processed by the PTMs and then do an aggregation over chunks.

Two Categories of Passage Aggregation Methods

- 1) **Passage Score Aggregation**: aggregate the relevance score of the query and segmented passage.
- 2) **Passage Representation Aggregation**: aggregate the representations of segmented passages to document representations first and then compute the relevance between query and the aggregated document representations.

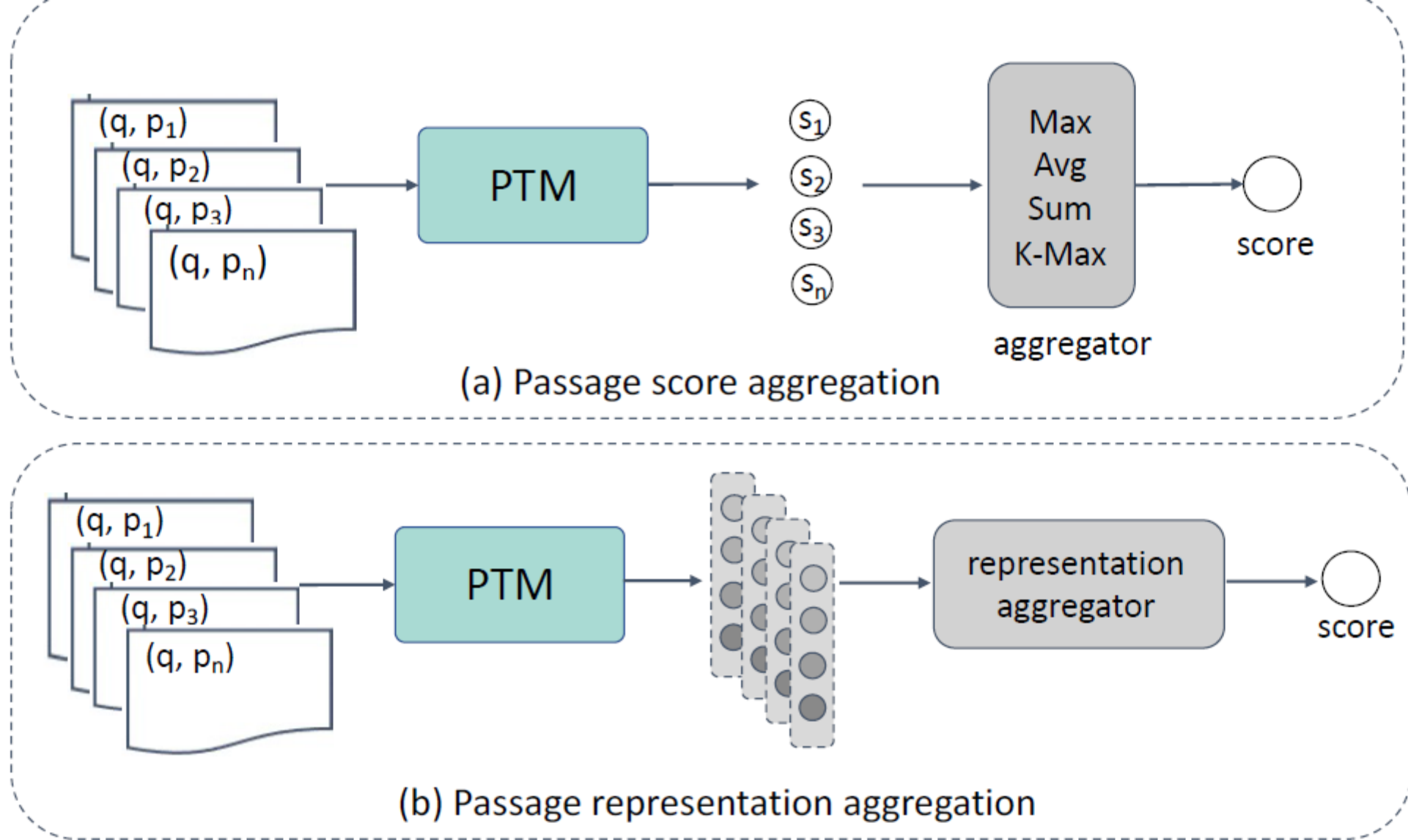
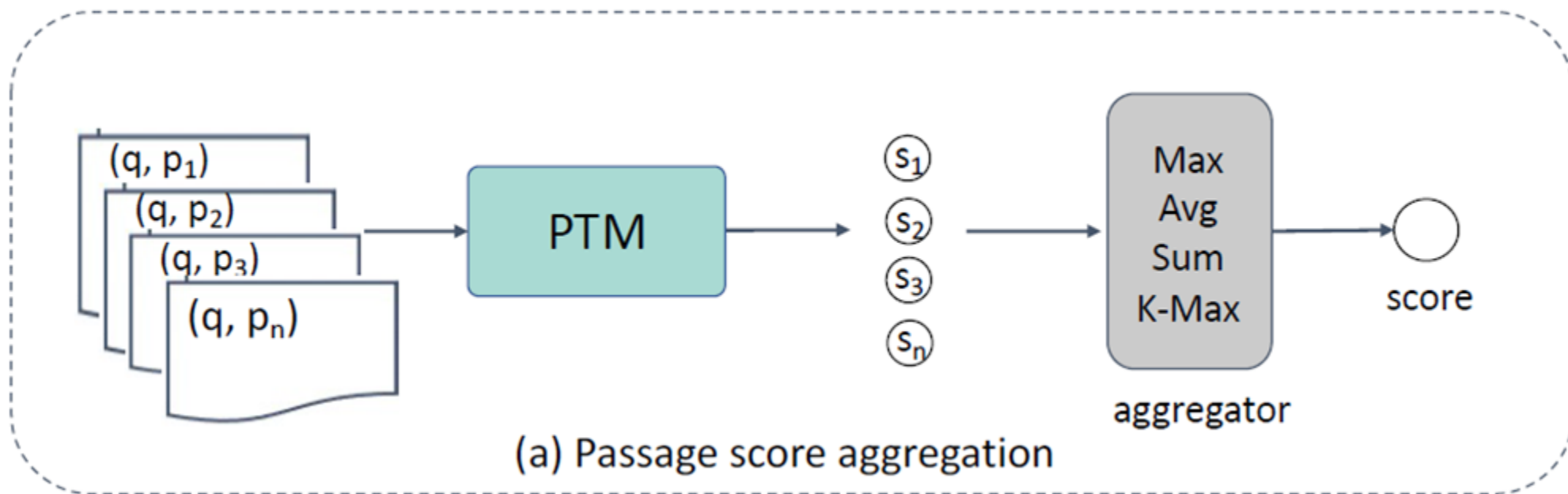


Figure 4.4: Two categories of passage aggregation methods.

Passage Score Aggregation

- A postprocessing method that only aggregates the relevance score between the query and the segmented passages provided by the PTMs.



Dai and Callan (2019b) Method

- Splitting a document into overlapping passages using a 150-word sliding window.
- Each passage will be concatenated with the query to input to the BERT, and the relevance of each passage is predicted independently.
- Three methods are proposed to aggregate the relevance scores of passages:
 - 1) BERT-firstP that only uses the score of the first passage
 - 2) BERT-maxP that uses the maximum score of the passages
 - 3) BERT-sumP that sums all the relevance scores of passages

Passage Representation Aggregation

- The relevance score is estimated by considering all the passages together
- PARADE
 - Segment the long document into a fixed number of overlapping chunks using 225-word sliding windows.
 - All passage representations from a document are aggregated for estimating the document relevance score
 - Using a mathematical operation such as the elementwise mean, max and sum on the representation vectors
 - Using a deep neural network including MLP, convolutional neural networks and Transformer layers

Model Acceleration

- The Transformer-based PTMs often consist of a tremendous amount of parameters ranging from millions to billions
- It's hard to deploy these PTMs on the online service in real-world applications or on resource restricted devices considering their requirement of low latency
- Methods
 - Decoupling the interaction of the query and the document
 - model distillation
 - dynamic modeling
 - lightweight fine-tuning

Decouple the Interaction

- Interaction-focused ranking models vs. Representation focused models
 - In the re-ranking stage, the interaction-focused ranking models that apply Transformer-based PTMs are widely-used and more effective than representation-focused ranking models.
 - The representation focused models are more efficient as they can pre-compute the document representations to reduce the online inference time.
- How to incorporate the advantage of the representation-focused architectures into the interaction-focused architectures?

PreTTR (MacAvaney et al., 2020)

- Employed the BERT model and proposed decoupling the low-level interaction of the query and the document via encoding them separately and then interacting **in the late BERT layers**.
- The document representations can be pre-computed offline and only the query needs to be encoded online.
- Most computational budget comes from the interaction of the last few layers.
- When merging the query representation and document representation on layer 11 of BERT, PreTTR achieved a 42X speedup on TREC Web-Track while not significantly reducing the ranking performance.
- Merging the representation of queries and documents at which layer depends on the datasets.

Model Distillation

- Knowledge distillation: method for reducing the computational cost by transferring knowledge from the teacher to the student
- Learn a smaller model from the outputs of a larger teacher model
- Gao et al. distill BERT for ranking
 - The teacher model uses BERT-base which contains 12 layers of Transformer, and the student model uses a 4 or 6 layers of Transformer
 - Only distilling the ranking information of the search task (Ranker Distill)
 - Distilling the MLM information over a large text corpus followed by a normal fine-tuning on the search task (LM Distill+Fine-tuning)
 - Distilling both (LM Distill+Ranker Distill)
- Distilling more knowledge from the teacher model can also benefit the ranking

Dynamic Modeling

- Adapt the model structures or parameters to different inputs is another promising method that can improve the efficiency of big models
- Selectively activate some model components of the whole model, such as some layers or a sub-network, conditioned on different inputs, and thus allocate computations on demand at the inference stage
- Early exit is a representative method in this line of research, which allows the examples to exit at early layers of the model without passing through the entire model.

Lightweight Fine-tuning

- Lightweight fine-tuning strategies update only a small number of extra parameters of PTMs while keeping most pre-trained parameters frozen.
- The intuitive method of lightweight fine-tuning is to freeze some or all pre-trained parameters.
- Insert small neural modules into existing models and only these inserted modules are fine-tuned on the downstream task.
- Jung et al. (2021) examined the lightweight fine-tuning methods in the PTMs-based ranking models.

Pre-training Methods Applied in Other Components

Components of a Search System

- Query Processing
 - Query Expansion
 - Query Rewriting
- User Intent Understanding
 - Query Suggestion
 - Search Clarification
 - Personalized Search
- Document Summarization
 - Generic Document Summarization
 - Snippet Generation
 - Keyphrase Extraction

Query Expansion

- Deal with the vocabulary mismatch problem or to mitigate the gap between queries and documents
 - Expanding the original query with the pre-trained word embeddings
 - Use word embeddings to incorporate and weight terms that are semantically similar to the query terms and further describe two query expansion models which are based on embeddings
 - Leverage the effective pseudo feedback-based relevance model for the expansion of the original query
 - BERT-QE
 - Leverages BERT as the backbone network to expand queries through three phases:
 - i) rerank candidate documents
 - ii) select relevant text chunks from the top-ranked documents to expand queries
 - iii) rerank the selected expansion chunks.
- These chunks will then be concatenated with the original queries for relevance scoring.

Query Rewriting

- 1) Map long-tail queries or questions into popular or frequent ones
- 2) Reformulate ambiguous input queries into well-formed queries to improve retrieval performance
- Conversational search
 - Utilize traditional IR query reformulation techniques to realize historical query expansion (HQE)
 - Then applied the T5-base model for neural transfer reformulation (NTR), i.e., rewriting a raw utterance into a natural language question without coreference and omission

Query Rewriting

- Match user queries or questions to Frequently Asked Questions (FAQs)
 - First employ BERT to calculate the semantic similarity between a query and the candidate FAQs
 - Further generate question candidates by fine-tuning GPT-2 in a well-designed unsupervised process and then filtered some noisy candidates according to the semantic similarity
- In dialogue systems, to simplify the multi-turn dialogue

User Intent Understanding

- Users may interact with the search system for multiple rounds
 - Search systems should understand users' evolving intent to better satisfy their information needs.
 - Besides modeling users' short-term intent with historical signals, the system can also forwardly provide assistance for search users.
- Related tasks
 - Query suggestion
 - Search clarification
 - Personalized search

Query Suggestion

- Provide users with possible future query options, aiming to help users complete their search tasks with less effort in complex search scenarios, e.g., session search or conversational search.
- Propose MeshBART which leverages user behavioral pattern such as clicks for generative query suggestion
- Conversational search
 - DeepSuggest finetunes BERT to rank question candidates by jointly optimizing four learning objectives
 - DeepSuggest-NLG adopts GPT-2 to generate question suggestions based on the maximum log-likelihood training

Search Clarification

- Query suggestions are usually presented in a post-search manner
- Systems can proactively ask user questions to clarify their information needs and reduce the uncertainty before returning the result list
- Utilize low-dimensional word embeddings learned by word2vec to clarify questions asked by users during a meeting
- Propose BERT-LeaQuR to encode both a query as well as its corresponding candidate questions and then employed a module called NeuQS to select high-quality clarifying questions
- Guided Transformer (GT), which utilizes external information such as the top retrieved documents and clarifying questions to learn better representations of input sequences by optimizing a multi-task learning objective
- Combined BERT with the maximum marginal-relevance (MMR) criterion to clarify user intents with fewer questions as possible

Personalized Search

- Provide personalized search services by modeling individual preferences in appropriate scenarios
- Encode the search history to capture the user's long-term and short-term interests
- Realize the personalized query expansion with the word embeddings learned on the user's profile
- Using personalized word embeddings can slightly improve the performance of E-mail search
- E-mail search requires personalization in conditions such as recency, user occupation, recipients, and attachments while protecting user privacy
- Leverage Transformer layers to encode personal E-mail search history, which only contains pre-processed features extracted from raw query and document text

Document Summarization

- Generic Document Summarization
 - Compressing given documents into a piece of concise text while keeping salient information
 - Extractive summarization and abstractive summarization
- Snippet Generation
 - Search snippets should highlight relevant points in the context of a given query
 - Extractive approaches and abstractive approaches

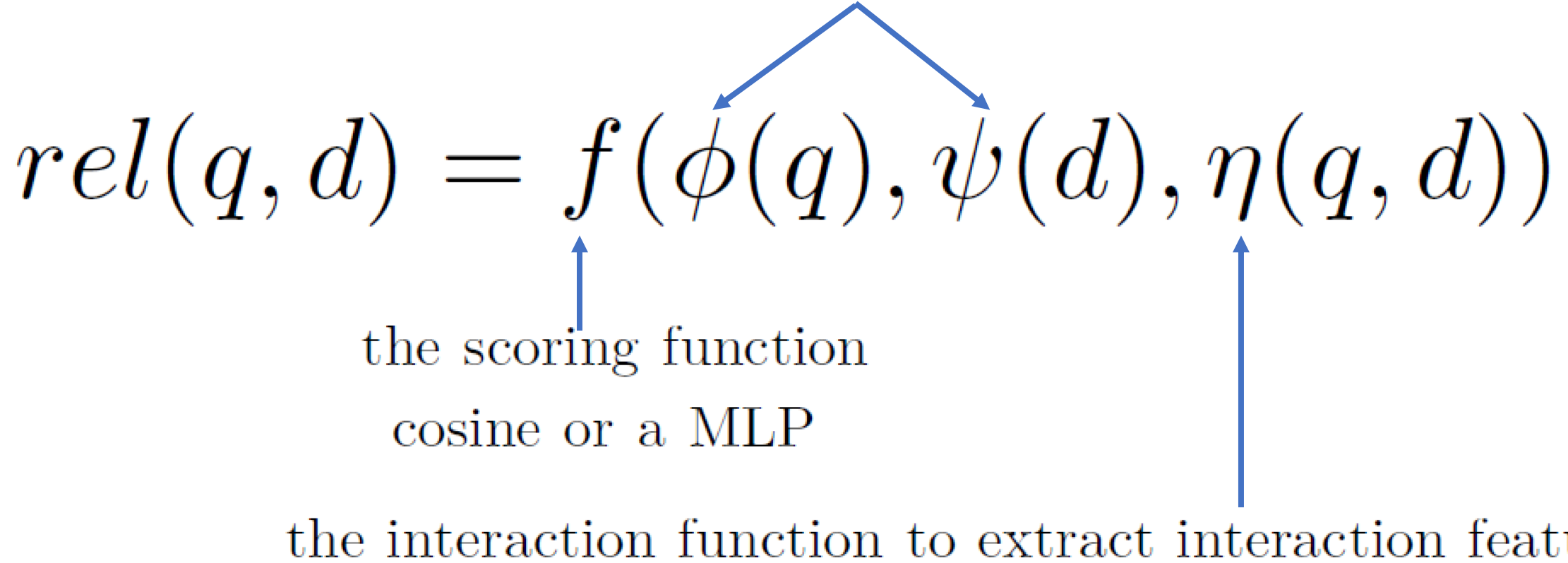
Pre-training Methods Designed for IR

Pre-training objectives and architectures from the IR perspectives

- Initially, PTMs were designed for NLP and the goal is to learn good representations for words or texts.
- When applying original PTMs in IR, studies have demonstrated that they can also benefit many IR tasks.
- The core of IR is to model the notion of relevance, which is not considered in the existing PTMs designed for NLP.
- Exploring new pre-training objectives as well as architectures from the IR perspectives.

General Ranking Function

representation functions to extract representation features


$$rel(q, d) = f(\phi(q), \psi(d), \eta(q, d))$$

The diagram illustrates the components of the general ranking function. A blue line from the text 'representation functions to extract representation features' branches into two arrows pointing to $\phi(q)$ and $\psi(d)$. A blue arrow points from the text 'the scoring function cosine or a MLP' to the function f . Another blue arrow points from the text 'the interaction function to extract interaction features' to $\eta(q, d)$.

the scoring function
cosine or a MLP

the interaction function to extract interaction features

Types of PTMs

- 1) Pre-training Embeddings Representation Models for IR
 - Take a single text sequence as input and learns contextualized word representations with various language modeling tasks
- 2) Pre-training Interaction Models for IR
 - Take a text sequence pair as input to directly learns their interactions
- The pre-trained representation models can also be applied to the interaction-focused architecture by fine-tuning on labeled data, and vice versa.

Pre-training Embeddings/Representation Models for IR

- Pre-trained word embeddings are mainly used to initialize the word embedding layer of a neural ranking model.
- Pre-trained representation models can be fully “transferred” to IR tasks without designing additional model architectures.
- Typical static embedding methods designed for NLP are trained based on word co-occurrence, especially the word proximity, in a large corpus.
- The main objective of IR is to predict the words observed in the documents relevant to a particular information need.

Word Embedding Methods Tailor for IR

- 1) Regularizing the Original Loss towards IR characteristics
 - Some IR-specific characteristics are not considered in the typical word embeddings designed for NLP, such as document-level word frequency and text length.
 - Including idf-based negative sampling, introducing L2 to regularize document length, and adding another objective for learning paradigmatic relations.
- 2) Designing New Objectives to capture relevance
 - Training local word embeddings in a query-specific manner that is, using query and the top k documents retrieved by a statistical language model approach to capture the nuances of topic-specific language.
 - This model needs to be trained during the query time and thus is not always practical in real-world applications.

Resources of Pre-training Methods in IR

- Datasets for Pre-Training
 - **Large collection size**: In a broad sense, collection size is a necessity for pre-training tasks in any deep learning fields.
 - **Structured documents**: The structures of a document include title, passages, sub-title, html structure, entity extractions, etc.
 - The second property are not always necessary for IR pre-training tasks.
 - These structures can be exploited in IR pre-training tasks to capture inter-page semantic relation.
 - Hyperlinks between the pages (e.g., anchor-page linking and page-page linking) provide intra-page semantic relations.

Table 7.1: Public available datasets which are potential for pre-training tasks.

Dataset	Source	#Docs	Language	Latest crawl date
Books ¹	Book	74M	ENG	2015
C4 ²	Web extracted text	0.3B	ENG	2019
Wikipedia ³	Wiki text	10M	Multi-lang	Monthly update
RealNews ⁴	News	120GB	ENG	2019
Amazon ⁵	Reviews	11GB	ENG	2003
WT10G ⁶	Web pages	1.7M	ENG	1997
GOV2 ⁷	Pages in .GOV	25M	ENG	2004
SogouT ⁸	Sogou web pages	1.17B	CHN	2016
ClueWeb09 ⁹	Web pages	1.04B	Multi-lang	2009
ClueWeb12 ¹⁰	Web pages	0.73B	ENG	2012
MS MARCO ¹¹	Bing web pages	3.2M	ENG	2018

¹<https://github.com/huggingface/datasets/tree/master/datasets/bookcorpus> Date accessed: 23 June 2022.

²<https://github.com/huggingface/datasets/tree/master/datasets/c4> Date accessed: 23 June 2022.

³<https://dumps.wikimedia.org/> Date accessed: 23 June 2022.

⁴<https://github.com/rowanz/grover/tree/master/realnews> Date accessed: 23 June 2022.

⁵<https://snap.stanford.edu/data/web-Amazon.html> Date accessed: 23 June 2022.

⁶http://ir.dcs.gla.ac.uk/test_collections/wt10g.html Date accessed: 23 June 2022.

⁷http://ir.dcs.gla.ac.uk/test_collections/access_to_data.html Date accessed: 23 June 2022.

⁸<http://www.sogou.com/labs/resource/t.php> Date accessed: 23 June 2022.

⁹<https://lemurproject.org/clueweb09/> Date accessed: 23 June 2022.

¹⁰<https://lemurproject.org/clueweb12/> Date accessed: 23 June 2022.

¹¹<https://microsoft.github.io/msmarco/> Date accessed: 23 June 2022.

General text corpus

- Widely used in NLP research for different tasks in different domains
- **Books**: This dataset aims to align books with the corresponding movie releases by associating the visual information with descriptive text.
- **C4**: Colossal Clean Crawled Corpus (C4) is a dataset consisting of more than 300 GBs clean English text scraped from the web.
- **Wikipedia**: Wikipedia is a large-scale collection containing all Wikimedia wikis in the form of wikitext source and metadata in XML structure.
- **RealNews**: RealNews is a large-scale corpus containing news articles from Common Crawl.
- **Amazon Reviews**: This dataset consists of Amazon shopping reviews from amazon.

IR related corpus

- Contain documents which are similar to downstream IR tasks
- WT10G (Web Track 10 Gigabytes: collected by CSIRO in Australia
- GOV2: a crawl of .gov sites in early 2004 which includes html, text and the extracted text of pdf, word and postscript.
- SogouT: a Chinese web page collection, which is sampled from Sogou search engines' indexed documents by considering both quality and diversity.
- Clubweb: a large-scale web document collection provided by CMU.
- MS MARCO: a popular large-scale document collection consisting of 3.2 million available documents, which are from the Bing search engine.

Datasets for Fine-Tuning

- Document-oriented
 - First stage retrieval (FSR): Retrieval stage from the full collection.
 - Ad-hoc ranking (AR): Ranking a candidate list given a query.
 - Session search (SS): Ranking a candidate list given a query and historical interactions.
 - Multi-modal ranking (MMR): Given a query, rank the candidate list where each item contains multiple heterogeneous information such as text, picture and html structure.
 - Personalized Search (PS): User-specific Ranking.

Datasets for Fine-Tuning

- Query-oriented
 - Query reformulation (QR): Iteratively modifying a query to improve the quality of search engine results in order to enhance user's search satisfaction.
 - Query suggestion (QS): Providing a suggestion which may be a reformulated query to better represent a user's search intent.
 - Query clarification (QC): Identifying user's search intent during a session.
- Others
 - Document summarization (DS): The process of shortening a document to create a subset (or a summary) that represents the most important information in this document.
 - Snippet generation (SG): Query-specific document summarization.
 - Keyphrase extraction (KE): It is also known as Keyword Extraction, which aims to automatically extract the most used and most important terms in a document.

Tasks

- 1. Robust track is a classic ad-hoc retrieval task in TREC which focuses on poorly performing topics. The released annotated collection only includes 250 queries and 50 queries in Robust04 and Robust05, respectively. This collection is used for evaluation in most experimental settings.
- 2. TREC Million Query (MQ) Track conducts an ad-hoc retrieval task over a large-scale collection of queries and documents. The final released dataset contains a four-level relevance judgement for each query-document pair.
- 3. Clueweb is another large-scale web search dataset provided by CMU. The “Category B” data set consists of the English pages, which is roughly the first 50 million pages of the entire data set.
- 4. TREC web track exploits the documents from Clueweb. The goal is to explore and evaluate specific aspects of Web retrieval, including the traditional ad-hoc retrieval task, risk-sensitive task and diversity search task.

Tasks

- 5. TREC Deep Learning Track studies IR in a large training data regime. It contains two tasks: Passage ranking and document ranking.
- 6. AOL is a publicly available query log released by the internet company AOL. The collection contains the query session, anonymized user ids and clicked documents, which are suitable for ad-hoc ranking, session search ranking, personalized search ranking, query reformulation and suggestion.
- 7. Sogou-QCL, Sogou-SRR (Search Result Relevance) and Tiangong-ST datasets were created from the Sougou search engine to support research on IR.

Table 7.2: Datasets for different downstream tasks in IR. Abbreviations in potential tasks are detailed in Section 7.2.

Dataset	Subdata	Size	Source	Potential Tasks
Robust	Robust04 Robust05	0.5M docs, 250 queries 1M docs, 50 queries	TREC Robust Track	FSR, AR, QR
TREC MQ	MQ2007 MQ2008	6.5K docs, 1.7K queries 1.4K docs, 784 queries	TREC Million Query track	FSR, AR, QR
Clueweb	09-CatB 12-CatB	50M docs, 150 queries 50M docs	Web pages	FSR, AR, QR, KE
TREC web track	99-2014	See ¹³	TREC web track	FSR, AR, QR
TREC DL track	2019-2021	See ¹⁴	TREC Deep Learning track	FSR, AR
AOL	\	6M queries	AOL Query logs	AR,SS,PS,QR,QS
Sogou-QCL	\	9M docs, 0.5M queries	Sogou Query logs	AR, QR
Sogou-SRR	\	63K results, 6K queries	Sogou Query logs	AR, MMR, QR
Tiangong-ST	\	0.3M docs, 40K queries	Sogou Query logs	AR, SS, QR, QS
Qulac	\	10K question-answer pairs	TREC Web Track	AR, QR, QC
BEIR	7 IR tasks	Vary from tasks	Wiki, Quaro, Twittter, News and etc.	FSR,AR, etc.
MS MARCO	2019-20	1M queries, 8.8M passages, 3.2M docs	TREC Deep Learning Track	FSR, AR, QR
TREC CAR	\	30M paras, 2M queries	TREC Complex answer retrieval	AR, QR, KE
CNN / Daily Mail	\	0.3M docs	Human generated abstracts	DS
New York Times (NYT)	\	1.8M docs	News articles	DS
Debatepedia	\	1,303 debates	Debate key points	SG, DS
DUC	2001-07	300 clusters, See ¹⁵	Doc understanding conference	SG, DS
WIKIREF	\	0.3M samples	QFS benchmark	SG, DS

¹³ <https://trec.nist.gov/data/webmain.html> Date accessed: 23 June 2022.

¹⁴ <https://microsoft.github.io/msmarco/TREC-Deep-Learning.html> Date accessed: 23 June 2022.

¹⁵ <https://duc.nist.gov/data.html> Date accessed: 23 June 2022.

Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks

arXiv:2005.11401v4 [cs.CL] 12 Apr 2021

Retrieval-Augmented Generation (RAG)

- A general-purpose fine-tuning recipe — models which combine pre-trained parametric and non-parametric memory for language generation
- The parametric memory is a pre-trained seq2seq model and the non-parametric memory is a dense vector index of Wikipedia, accessed with a pre-trained neural retriever

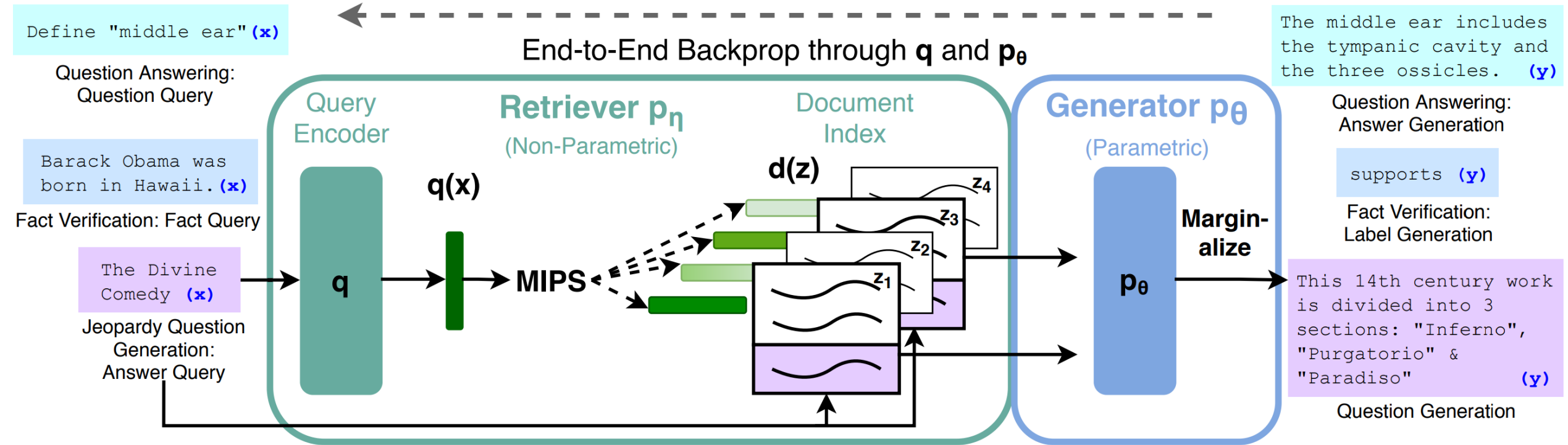


Figure 1: Overview of our approach. We combine a pre-trained retriever (*Query Encoder* + *Document Index*) with a pre-trained seq2seq model (*Generator*) and fine-tune end-to-end. For query x , we use Maximum Inner Product Search (MIPS) to find the top-K documents z_i . For final prediction y , we treat z as a latent variable and marginalize over seq2seq predictions given different documents.

Models

RAG-Sequence Model The RAG-Sequence model uses the same retrieved document to generate the complete *sequence*. Technically, it treats the retrieved document as a single latent variable that is marginalized to get the seq2seq probability $p(y|x)$ via a top-K approximation. Concretely, the top K documents are retrieved using the retriever, and the generator produces the output sequence probability for each document, which are then marginalized,

$$p_{\text{RAG-Sequence}}(y|x) \approx \sum_{z \in \text{top-}k(p(\cdot|x))} p_{\eta}(z|x) p_{\theta}(y|x, z) = \sum_{z \in \text{top-}k(p(\cdot|x))} p_{\eta}(z|x) \prod_i^N p_{\theta}(y_i|x, z, y_{1:i-1})$$

RAG-Token Model In the RAG-Token model we can draw a different latent document for each target *token* and marginalize accordingly. This allows the generator to choose content from several documents when producing an answer. Concretely, the top K documents are retrieved using the retriever, and then the generator produces a distribution for the next output token for each document, before marginalizing, and repeating the process with the following output token, Formally, we define:

$$p_{\text{RAG-Token}}(y|x) \approx \prod_i^N \sum_{z \in \text{top-}k(p(\cdot|x))} p_{\eta}(z|x) p_{\theta}(y_i|x, z, y_{1:i-1})$$

Retriever: DPR

The retrieval component $p_\eta(z|x)$ is based on DPR [26]. DPR follows a bi-encoder architecture:

$$p_\eta(z|x) \propto \exp(\mathbf{d}(z)^\top \mathbf{q}(x)) \quad \mathbf{d}(z) = \text{BERT}_d(z), \quad \mathbf{q}(x) = \text{BERT}_q(x)$$

where $\mathbf{d}(z)$ is a dense representation of a document produced by a $\text{BERT}_{\text{BASE}}$ *document encoder* [8], and $\mathbf{q}(x)$ a query representation produced by a *query encoder*, also based on $\text{BERT}_{\text{BASE}}$. Calculating $\text{top-k}(p_\eta(\cdot|x))$, the list of k documents z with highest prior probability $p_\eta(z|x)$, is a Maximum Inner Product Search (MIPS) problem, which can be approximately solved in sub-linear time [23]. We use a pre-trained bi-encoder from DPR to initialize our retriever and to build the document index. This retriever was trained to retrieve documents which contain answers to TriviaQA [24] questions and Natural Questions [29]. We refer to the document index as the *non-parametric memory*.

Generator: BAR

The generator component $p_{\theta}(y_i|x, z, y_{1:i-1})$ could be modelled using any encoder-decoder. We use BART-large [32], a pre-trained seq2seq transformer [58] with 400M parameters. To combine the input x with the retrieved content z when generating from BART, we simply concatenate them. BART was pre-trained using a denoising objective and a variety of different noising functions. It has obtained state-of-the-art results on a diverse set of generation tasks and outperforms comparably-sized T5 models [32]. We refer to the BART generator parameters θ as the *parametric memory* henceforth.

Nonparametric Decoding for Generative Retrieval

arXiv:2210.02068v3 [cs.IR] 28 May 2023