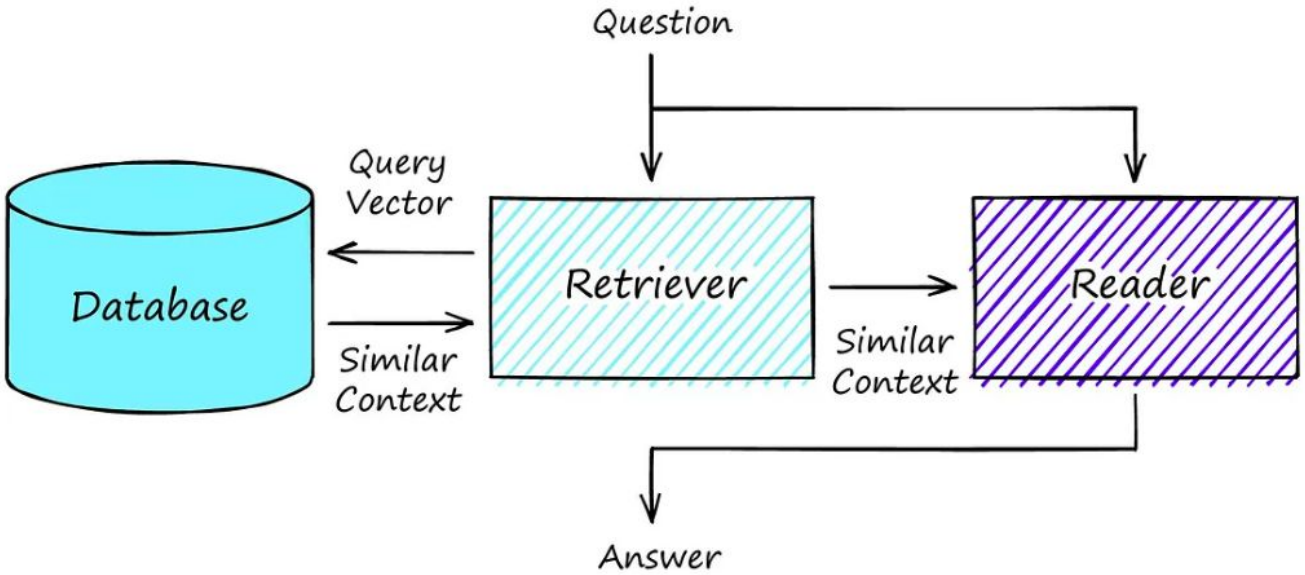
Independent Study S’24

Advisor: Dr Manish Shrivastava
By: Hitesh Goel (2020115003)

Retriever-reader (reranker approaches)



Overview of a Reader-Retriever System. Source: https://www.pinecone.io/learn/question-answering

Retriever retrieves top k<n documents. Reader either directly answers from this context or we can also rerank the k documents.

Retriever survey

DPR

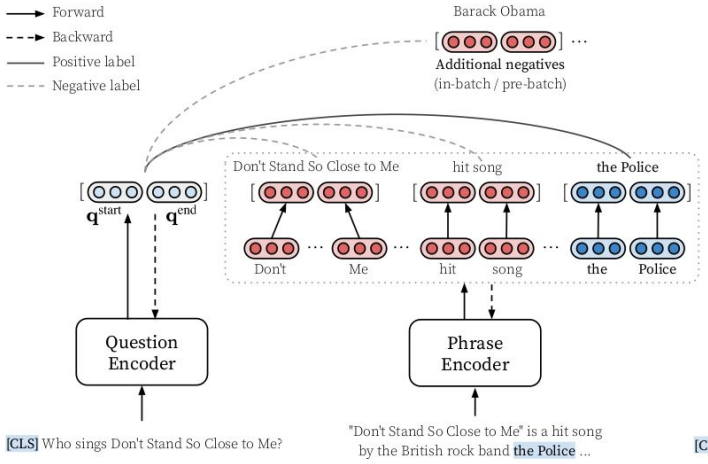
$$\text{sim}(q, p) = E_Q(q)^\intercal E_P(p).$$

Let $\mathcal{D} = \{\langle q_i, p_i^+, p_{i,1}^-, \cdots, p_{i,n}^- \rangle\}_{i=1}^m$ be the training data that consists of m instances. Each instance contains one question q_i and one relevant (positive) passage p_i^+, along with n irrelevant (negative) passages $p_{i,j}^-$. We optimize the loss function as the negative log likelihood of the positive passage:

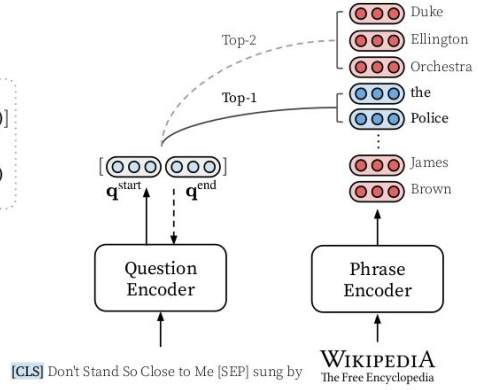
$$L(q_i, p_i^+, p_{i,1}^-, \cdots, p_{i,n}^-) \quad (2)$$
$$= -\log \frac{e^{\text{sim}(q_i, p_i^+)}}{e^{\text{sim}(q_i, p_i^+)} + \sum_{j=1}^n e^{\text{sim}(q_i, p_{i,j}^-)}}.$$

- A dual encoder: query and passage
- Cosine similarity used as the distance metric
- Contrastive loss to learn (negative sampling)

Densephrases



(a) Single-passage training w/ additional negatives



(b) Query-side fine-tuning & Inference

- Phrase encoder and question encoder
- Freeze the phrase enc and store all embeddings offline for efficient search (only need to store start and end token/indices)
- Query side finetuning for further improvements.

GNN Encoder

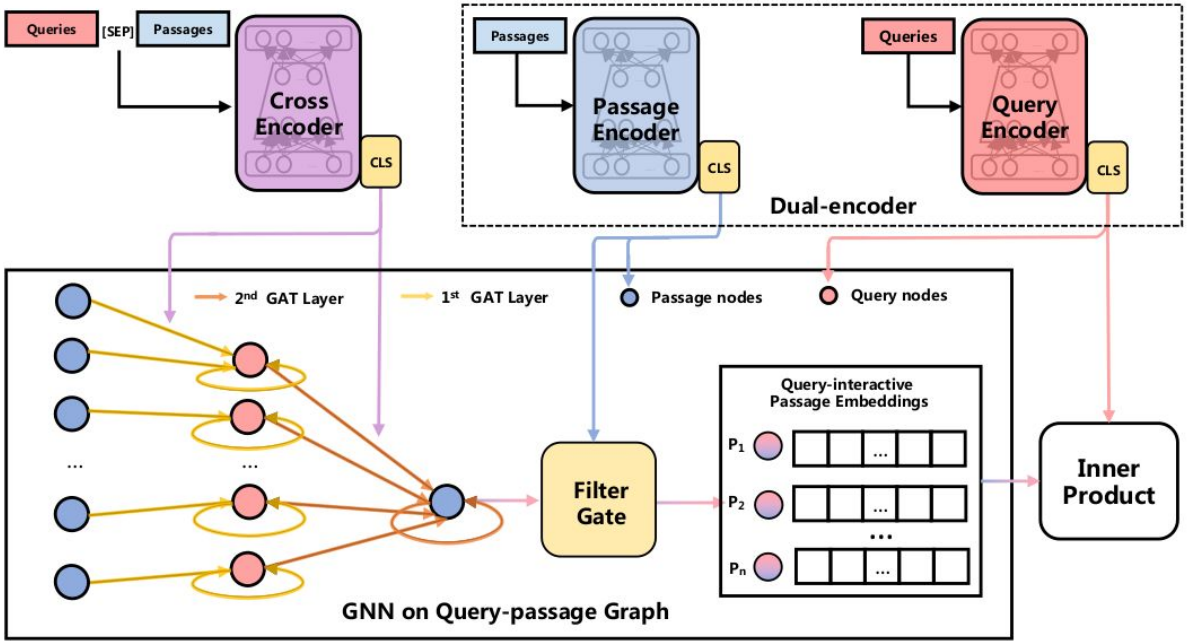


Figure 1: Overview of GNN-encoder which can be divided into three parts: (1) Dual-encoder; (2) Cross-encoder; (3) GNN on query-passage graph. Node and edge features of query-passage graph are initialized by dual-encoder and cross-encoder, respectively. Only the parameters of dual-encoder and GNN will be updated during training.

- A dual enc loses interaction between query-passage pair
- Query info is fused into passage representations (and vice versa) using GNNs
- Cross encoder (GNN) + dual encoder

SimLM

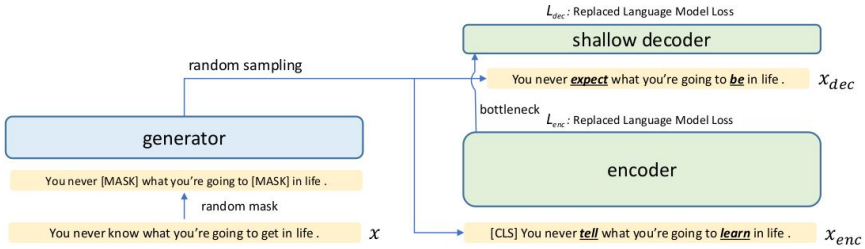


Figure 1: Pre-training architecture of SimLM. Replaced tokens (underlined) are randomly sampled from the generator distribution.

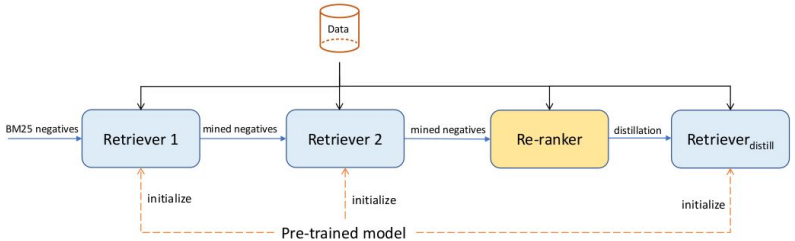


Figure 2: Illustration of our supervised fine-tuning pipeline. Note that we only use SimLM to initialize the biencoder-based retrievers. For cross-encoder based re-ranker, we use off-the-shelf pre-trained models such as $\text{ELECTRA}_{\text{base}}$.

- Better performance than GNNs
- They use a representation bottleneck (shallow decoder)
- Using this bottleneck architecture, the model learns to compress passage info into a dense vector through self-supervised pretraining.
- Good performance as retriever, not so much as a reranker.

ART (Questions are all you need)

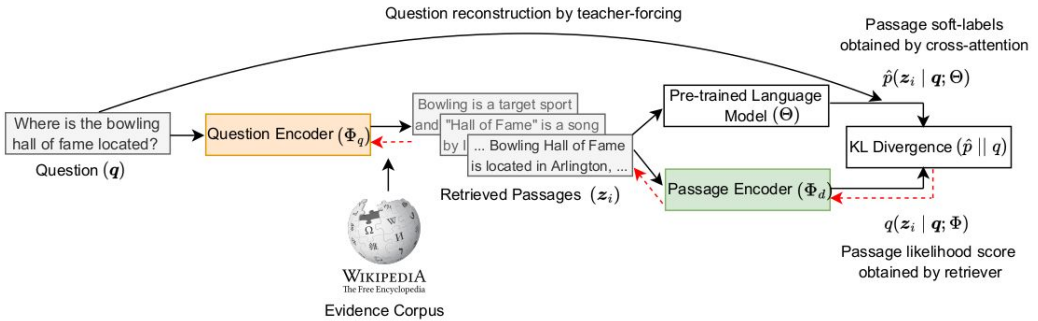
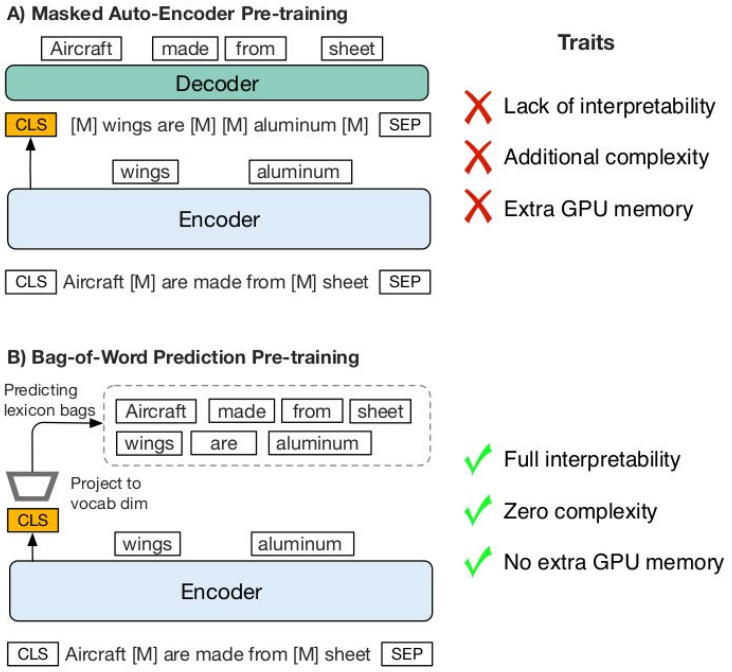


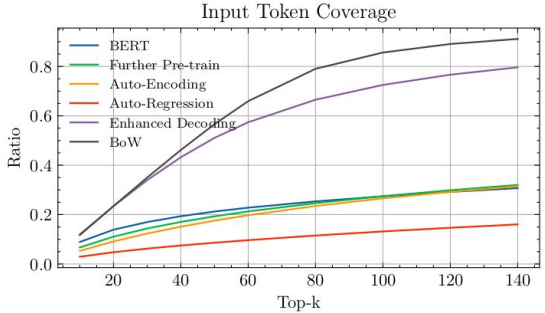
Figure 1: ART maximizes the retrieved passage likelihood computed from the dense retriever by considering the language model question reconstruction score conditioned on the passage as a *soft-label*. Colored blocks indicate trainable parameters. Red arrows show gradient flow during backpropagation.

$$\log p(\boldsymbol{z} \mid \boldsymbol{q}; \Theta)$$
$$= \log p(\boldsymbol{q} \mid \boldsymbol{z}; \Theta) + \log p(\boldsymbol{z}) + c \quad (2a)$$
$$\propto \frac{1}{|\boldsymbol{q}|} \sum_t \log p(q_t \mid \boldsymbol{q}_{<t}, \boldsymbol{z}; \Theta), \quad (2b)$$

- Unsupervised method
- Auto-encoding based – reconstructs original question
- We use log(q|z) to estimate likelihood of log(z|q) – simple Bayes rule.
- T0 (3B) outperforms instruction-tuned T5-lm-adapt.
- Scaling T0 (3B) to 11B does not result in any improvements

Drop your decoder: Pretraining with Bag of words





$$h_{proj} = h_{last}^{CLS} \cdot E^T$$

heads (i.e., its embedding matrix E^I). Then we apply a multi-label cross-entropy loss to predict the input token set T directly. Bag-of-Word means the label T is a set of tokenized input text. For simplicity, it is not related to positional information, similar to the traits of sparse retrieval [26].

$$\mathcal{L}_{Dec} = CE(h_{last}^{CLS} E^T, T) \quad (12)$$

Figure 1: Comparison of Masked Auto-Encoder Pre-training and Bag-of-Word Prediction Pre-training.

BoW (continued)

- revealing that masked auto-encoder (MAE) pre-training with enhanced decoding significantly improves the term coverage of inp tokens in dense representations, compared to vanilla BERT checkpoints.
- we propose a modification to the traditional MAE by replacing the decoder of a masked auto-encoder with a completely simplified Bag-of-Word prediction task.
- state-of-the-art retrieval performance on several large-scale retrieval benchmarks without requiring any additional parameters, which provides a 67% training speed-up compared to standard masked auto-encoder pre-training with enhanced decoding.
- issues with MAE style trained retrievers
    - difficult to interpret why they work
    - MAE-style pre-training mandatory needs the additional Transformers-based decoders, which brings considerable GPU memory cost and additional $O(n^2)$ computational complexity

Results

	MSMarco		NQ	
	NDCG@10	Recall@100	NDCG@10	Recall@100
DPR	17.7	55.2	41.5	88
Densephrases	-	-	40.9	84.7
GNN encoder	-	86.9	-	88.5
SimLM*	**34.3**	**92.9**	48.7	**88.7**
ART*	32.1	80.9	39.8	87.4
BoW*	-	-	**50.04**	88.6

Other papers explored

- Contrastive span prediction (Costa)
- Deep prompt tuning
- Angle Optimised text embeddings
- In-context retrieval augmented LMs
- RAG: Is Dense Passage Retrieval Retrieving
- Abstract meaning representation for open-domain QA

Rerankers survey

Retrieve, Rerank and Generate

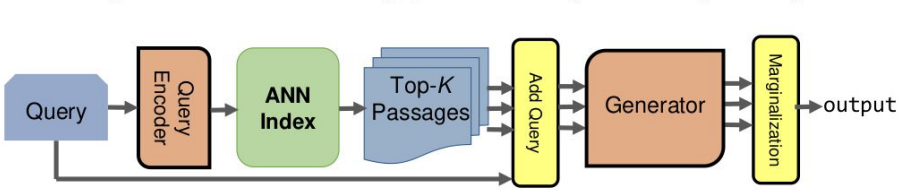


Figure 2: RAG Architecture

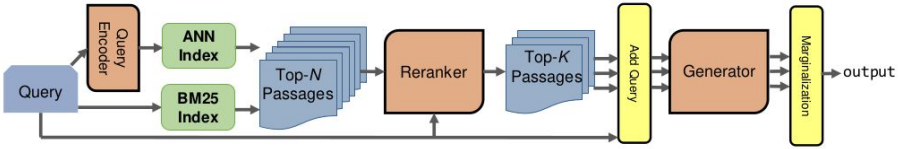


Figure 3: Re^{2}G Architecture

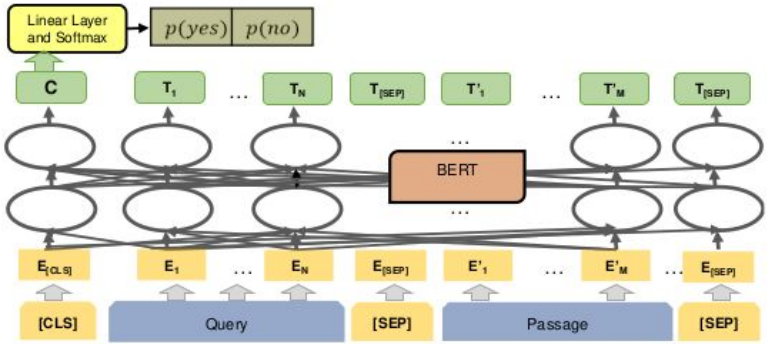


Figure 4: Interaction Model Reranker

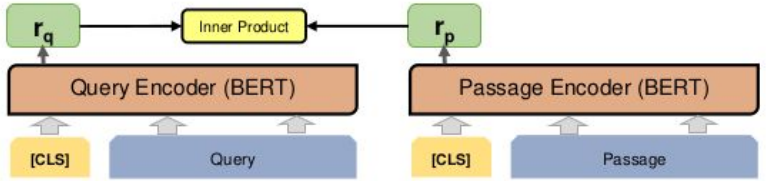


Figure 5: Representation Model for Initial Retrieval

Unsupervised Passage ranking

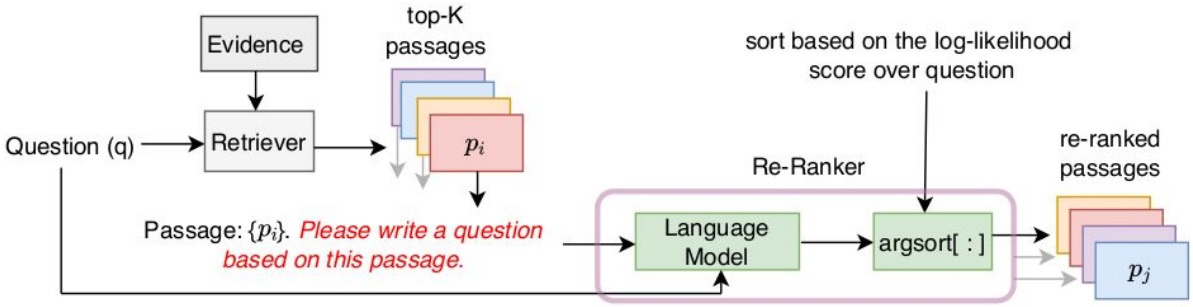


Figure 2: An illustration of the different components in UPR. For more details, please refer to text.

$$\log p(\boldsymbol{z}_i \mid \boldsymbol{q}) = \log p(\boldsymbol{q} \mid \boldsymbol{z}_i) + \log p(\boldsymbol{z}_i) + c \,,$$

where $p(\boldsymbol{z}_i)$ is the prior on the retrieved passage and c is a common constant for all $\boldsymbol{z}_i$.

As a simplifying assumption, we assume that the passage prior $\log p(\boldsymbol{z}_i)$ is uniform, and can be ignored for re-ranking. With this, the above expression reduces to

$$\log p(\boldsymbol{z}_i \mid \boldsymbol{q}) \propto \log p(\boldsymbol{q} \mid \boldsymbol{z}_i), \; \forall \boldsymbol{z}_i \in \mathcal{Z} \,.$$

We estimate $\log p(\boldsymbol{q} \mid \boldsymbol{z}_i)$ using a pre-trained language model (PLM) to compute the average log-likelihood of the question tokens conditioned on the passage:

$$\log p(\boldsymbol{q} \mid \boldsymbol{z}_i) = \frac{1}{|\boldsymbol{q}|} \sum_t \log p(q_t \mid \boldsymbol{q}_{<t}, \boldsymbol{z}_i; \Theta) \,.$$

- Bayes rule – estimate p(z|q) using p(q|z) by reconstructing the question.
- Cross attention helps attend each word in the passage.
- T0 (11B) works best, we run T0 (3B).

RankT5

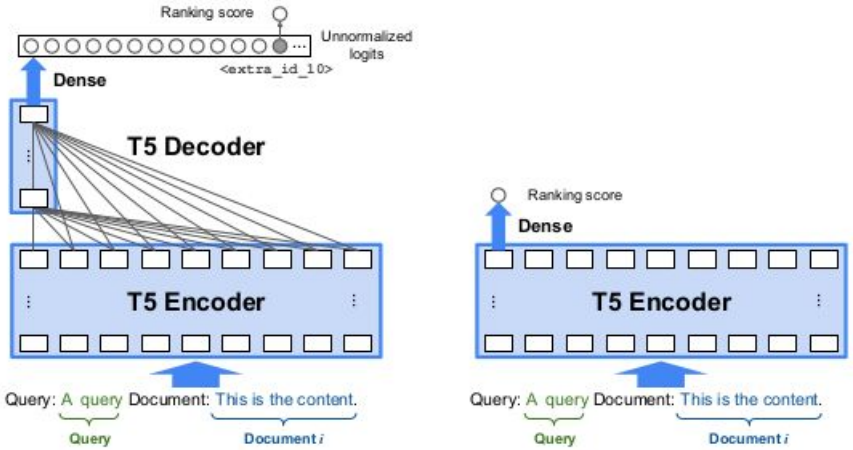
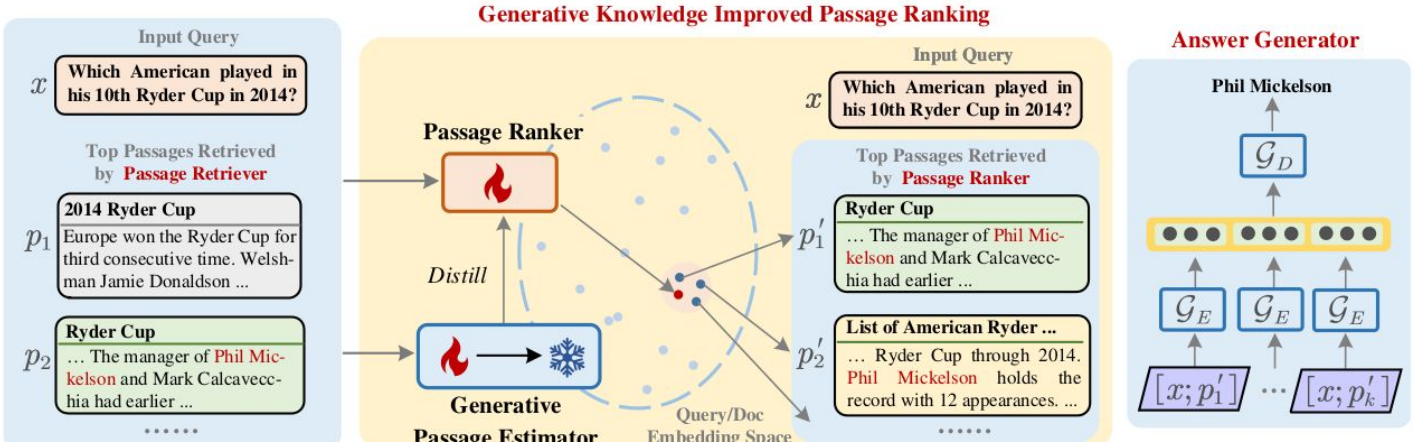


Figure 1: Model structures of the two variants of RankT5.

- Limited studies on how to leverage more powerful seq2seq models.
- Two T5 models: enc-dec and an only encoder model since auto-regressive decoding is not required.
- Finetuning with listwise ranking losses (ListMLE, etc).

GripRank



- by distilling knowledge from a generative passage estimator (GPE) to a passage ranker, where the GPE is a generative language model used to measure how likely the candidate passages can generate the proper answer.
- they train a generative passage estimator (GPE) under the supervision of the golden answer, taking the concatenation of the query and golden passage as input. Once the GPE finishes training, we freeze the entire GPE and the passage ranker learns to rank the candidate passages ordered by the GPE. Each top-ranked passage is then concatenated with the input query for answer generation.
- GPE is BARTLarge
- passage ranker is a cross encoder architecture. ElectraBase pretrained on MS Marco as the backbone.
- curriculum learning for knowledge distillation

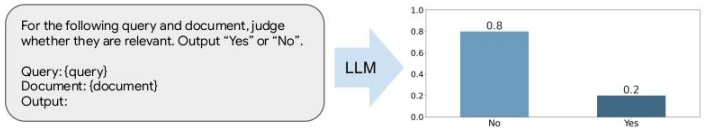
Open-source LLMs are zero-shot QLMs for document ranking

$$S_{QLM}(\boldsymbol{q}, d) = \frac{1}{|\boldsymbol{q}|} \sum_{t} \log \text{LLM}(q_t | \boldsymbol{p}, d, \boldsymbol{q}_{<t})$$

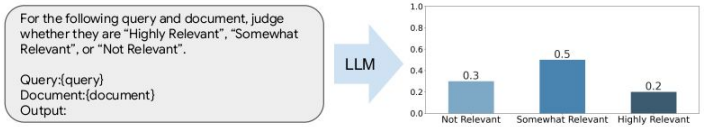
$$S(\boldsymbol{q}, d) = \alpha \cdot S_{BM25}(\boldsymbol{q}, d) + (1 - \alpha) \cdot S_{QLM}(\boldsymbol{q}, d),$$

- Query likelihood models: rank docs based on the probabilty of generating the query given the content of the document.
- Directly links to the UPE paper – they use T5 models, which have already been finetuned on downstream tasks. Hence that paper is more akin to transfer learning than zero-shot learning.
- Instead, they use LLMs like LLama-2.
- These models are not trained on QA datasets previously.
- Rerankers finetuned on QA datasets consistently out-perform zero-shot models. But the QLMs still display competitive performance.
- Instruction tuning can hinder performance because the models then start paying more attention to the task than the input text.

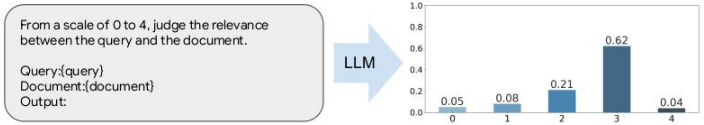
Beyond Yes and No: Improving Zero-Shot LLM Rankers via Scoring Fine-Grained Relevance Labels



(a) Yes-No relevance generation



(b) Fine-grained relevance label generation



(c) Rating scale relevance generation

$$f(q, d_i) = \frac{\exp(s_{i,1})}{\exp(s_{i,1}) + \exp(s_{i,0})}$$

- previously, prompts: pointwise ranking (yes/no answers to whether a passage can answer a question)
- the lack of intermediate relevance label options may cause the LLM to provide noisy or biased answers for documents that are partially relevant to the query
- paper proposes to incorporate fine-grained relevance labels into the prompt for LLM rankers, enabling them to better differentiate among documents with different levels of relevance to the query and thus derive a more accurate ranking.
- Instead of asking the LLM to choose between two options, we provide the LLM with fine-grained relevance labels, such as "Highly Relevant", "Somewhat Relevant" and "Not Relevant" and collect their likelihood scores from LLM predictions to derive the ranking score. The intuition is that the intermediate relevance labels in the prompt serve as a "cue" to the LLM to distinguish partially relevant documents from fully relevant or fully irrelevant ones.
- perf remains similar for k between 4 and 8. decreases if we inc k further (k=num of labels)
  - thus perf does not always inc with increasing labels

Generating Diverse Criteria On-the-Fly to Improve Pointwise LLM Rankers

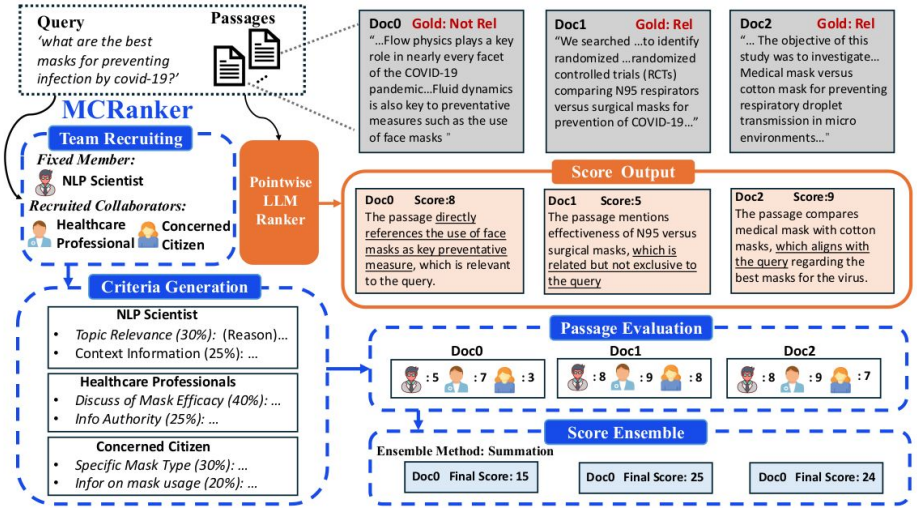


Figure 1: Pipeline of the proposed MCRanker in blue dashed line and the example output of the Pointwise LLM-based Ranker in orange straight line.

- 2 major drawbacks of pointwise rerankers
  - they fail to follow a standardized comparison guidance during the ranking process
  - they struggle with comprehensive considerations when dealing with complicated passages
- propose to build a ranker that generates ranking scores based on a set of criteria from various perspectives.
- This error happens when the LLM ranker decides to adopt a biased assessment criterion that prioritizes keyword presence over the nuanced understanding of content semantics.
- MCRanker emulates the domain expertise and text analytical capabilities of professional annotators through virtual team recruiting.
- 4 steps
  - team recruiting: nlp scientist and others
  - criteria gen
  - passage eval
  - score ensemble

Results

	NQ	TriviaQA
	R@5	
UPR*	64.7	64.4
Re2G	76.6	74.2
GripRank	77.6	78.8

Results

Model	BEIR datasets (Ndcg@10 average)
QG	0.45
Fine-grained relevance labels	0.50
Diverse criteria	0.51
Open-source LLMs*	0.43

Thank you

Questions and Feedback?