Efficient and Interpretable Information Retrieval for Product Question Answering with Heterogeneous Data

Biplob Biswas, Rajiv Ramnath
The Ohio State University
Columbus OH 43210, USA
{biswas.102, ramnath.6}@osu.edu

Abstract

Expansion-enhanced sparse lexical representation improves information retrieval (IR) by minimizing vocabulary mismatch problems during lexical matching. In this paper, we explore the potential of jointly learning dense semantic representation and combining it with the lexical one for ranking candidate information. We present a hybrid information retrieval mechanism that maximizes lexical and semantic matching while minimizing their shortcomings. Our architecture consists of dual hybrid encoders that independently encode queries and information elements. Each encoder jointly learns a dense semantic representation and a sparse lexical representation augmented by a learnable term expansion of the corresponding text through contrastive learning. We demonstrate the efficacy of our model in single-stage ranking of a benchmark product question-answering dataset containing the typical heterogeneous information available on online product pages. Our evaluation demonstrates that our hybrid approach outperforms independently trained retrievers by 10.95% (sparse) and 2.7% (dense) in MRR@5 score. Moreover, our model offers better interpretability and performs comparably to state-of-the-art cross encoders while reducing response time by 30% (latency) and cutting computational load by approximately 38% (FLOPs).

Keywords: Hybrid Information Retrieval, Interpretability, Heterogeneous Product Question-Answering

1. Introduction

In the field of natural language processing, ranked information retrieval (IR), refers to retrieving information ordered by relevance from a large collection, in response to a query. Ranked IR remains important even with the emergence of advanced large language models (LLMs) as a means of greatly enriching their outputs.

Existing retrieval approaches can be categorized into two groups - sparse and dense. Sparse retrieval uses a token-based sparse representation of the query and the information, such as bag-of-words (BoW) obtained via TF-IDF (Sparck Jones, 1988) or BM25 (Robertson and Walker, 1994), and an inverted index for query processing. Although these BoW models facilitate faster retrieval, they rely on exact matches, and hence cannot identify semantically relevant information having a different set of tokens than the query. Dense retrieval, on the other hand, retrieves by comparing dense representations often computed by neural networks such as BERT (Devlin et al., 2019). While these models can perform semantic-level matching, their computational complexity renders them impractical for online real-time ranking when the corpus becomes large.

In an effort to balance the quality-cost trade-off, a two-stage pipeline is proposed where a quicker retriever first retrieves a smaller set of candidates and then a dense retriever re-ranks them in a second stage. Unfortunately, this approach suffers from two major problems. First, any semantically relevant information pruned due to lack of exact word matches in the first stage is not considered for further ranking. Second, the neural ranker in the last stage lacks interpretability because, for scoring, it uses the inner product of the latent representation of the text which is difficult to explain in human understandable terms. Recently proposed transformer (Vaswani et al., 2017) encoders have the potential to tackle these issues. By utilizing a pre-trained masked language model (MLM), SparTerm (Bai et al., 2020) and SPLADE (Formal et al., 2021) progressively improved the use of expansion-aware sparse lexical representation learners in mitigating vocabulary mismatch problems, while enhancing interpretability. SparseEmbed (Kong et al., 2023) further extended this concept by learning contextual embeddings of the top-k tokens in the lexical representation. However, these models ignore the text-level dense representation (i.e. [CLS] token encoding) which captures the summarized expression of a text. Furthermore, being a byproduct of the BERT with MLM head, it can be obtained without additional computation and stored as a single vector. Finally, jointly learning lexical and semantic representations can pave the way for a single-stage ranking, especially in product-question-answering tasks (Shen et al., 2022) where information from an online product page can be pre-computed offline and then ranked at query time.

In this work, we investigate these possibilities and present a hybrid information ranker that balances the quality, cost, and interpretability by incorporating both lexical and semantic matching in ranking.

The contribution of our work is in two areas:

- We present a hybrid ranking model that jointly learns semantic and lexical representations and combines them for efficient information retrieval.
- We evaluate our model on a heterogeneous product question-answering dataset and show that our approach provides better performance and interpretability with a reasonable computational complexity and memory footprint. Our code is available online[1].

2. Related Works

Our hybrid model brings together ideas from both dense retrieval and sparse retrieval. Based on the scoring process, we find three variants of dense retrievers (as shown in Figure 1) related to our work; all of them employ pre-trained language models to learn dense semantic representations. Nogueira et al. (Nogueira and Cho, 2019) used BERT as a *cross-encoder*(Figure 1(c)) where concatenated query-information sequence is processed simultaneously through all-to-all interactions and a binary classifier maps the resultant representation to relevance probability. In DPR (Karpukhin et al., 2020), Karpukhin et al. employed two *independent dense encoders* (Figure 1(a)) that separately map query and information into their single-vector dense representations and the information score is computed by their inner product. To improve model expressiveness, Khattab et al. proposed ColBERT (Khattab and Zaharia, 2020), a *late-interaction*(Figure 1(b)) model, to utilize a multi-vector representation from dual encoders that allow deferred cross interaction among contextual token encodings. However, ColBERT suffers from scalability issues as it requires storing and indexing all the token encodings in a sequence.

Term-based BM25 (Robertson and Walker, 1994) has been long used as a baseline for sparse retrieval. In order to capture semantic relationships in sparse representations, SNRM (Zamani et al., 2018) uses high-dimensional vectors of latent terms. However, it loses the interpretability provided by actual vocabulary terms. SparTerm (Bai et al., 2020) addresses this (interpretability) issue by mapping text to a sparse term-importance distribution in BERT vocabulary space. In SPLADE (Formal et al., 2021), Formal et al. extended this idea by introducing a log-saturation effect in term-importance estimation and sparsity regularization in training loss. Following this, SparseEmbed (Kong et al., 2023) learns and uses contextual embeddings of the sparse lexical representation to improve model expressiveness. Our approach closely follows this direction of research. However, instead of only comparing lexical representation, we also consider summarized semantic matching without increasing encoding complexity, by leveraging the fact that BERT computes the [CLS] token encoding anyway. Moreover, unlike prior hybrid models (Karpukhin et al., 2020; Ma et al., 2021; Gao et al., 2021; Luan et al., 2021), our model *jointly* learns semantic representations and expandable lexical representations, enabling interpretability with expanded tokens.

[1] https://github.com/biplob1ly/HybridPQA

Evidence Ranking

Items	Train	Validation	Test
Total records	24295	2731	309347
Unique query	4528	509	2773
Mean candidates per query	5.37	5.37	111.56
Mean +ve candidate ratio	0.25	0.24	0.06
Mean question words	11.23	11.73	6.98
Mean candidate words	17.19	18.49	12.59
Mean sources per query	1.09	1.10	5.12

Answer Generation

Items	Train	Validation	Test
Total records	3693	398	2289
Unique query	3356	395	1340
Mean evidences per query	1.1	1.01	1.71
Mean answer words	8.22	8.27	7.24

Table 1: The summary of the hetPQA (Shen et al., 2022) dataset.

3. Dataset

We apply our model to hetPQA (Shen et al., 2022), a large-scale benchmark dataset for product question-answering systems, that provides various information from product web pages as candidate evidence to answer a product-specific query. In production, after ranking the candidate evidence elements for a query, the higher-ranked ones are utilized for answer generation. The information (evidence) is extracted from heterogeneous sources that include: 1. product attributes in JSON format, 2. bullet points from product summary, 3. community answers to product questions (CQA), 4. product descriptions, 5. on-site publications (OSP) about products, and 6. user reviews on the product page. The collection has separate sets of data for evidence ranking and answer generation, and each dataset comprises train, validation, and test split. The details of the splits are reported in Table 1. Further, our manual inspection of the BM25-driven evidence ranking result on the test set revealed 1377 incorrect annotations; these were corrected. We have disclosed our correction in the

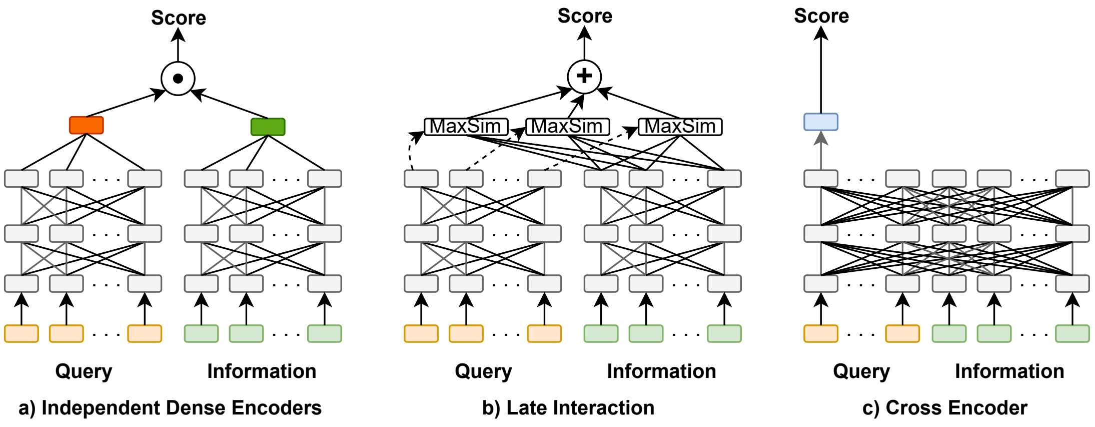

Figure 1: Existing neural rankers with different interaction schemes.

Items	Attribute	Bullet	CQA	Desc	OSP	Review
Ranking	4.0%	4.4%	44.1%	12.8%	2.6%	32.1%
Generation	11.4%	16.6%	21.8%	17.8%	8.6%	23.8%
Mean #words	5.8	12.6	13.3	12.9	17.8	18.4

Table 2: The distribution of sources and mean word count in the hetPQA (Shen et al., 2022) dataset.

repository shared above and also conducted all our experiments with the amended test set. Altogether, the evidence ranking set has 7585 unique questions and 149283 unique pieces of information distributed over the aforementioned 6 sources. The answer generation set contains a total of 5037 unique questions and 5229 unique evidence elements. The overall source distribution and average word counts are given in Table 2. More details can be found in hetPQA (Shen et al., 2022) paper.

Data Preparation To begin with, the text was normalized to a canonical representation. All non-English characters were replaced by their equivalents. Symbols and short forms of dimensions (e.g. $3'' \ l \times 4'' \ w$) were substituted by the corresponding English words (*length 3 inches* $\times$ *width 4 inches*). We also flattened JSON-formatted attributes to comma-separated strings.

4. Framework

Our framework comprises two major components: a ranker and a generator. Given a query and a set of candidate information, the ranker sorts the information in descending order of relevance. The generator then produces a coherent and informative response from the top-ranked results. We elaborate on this in the subsections below.

4.1. Ranker

The key function of a ranker is to measure the relevance of each candidate information element with respect to the query. Figure 2 depicts the architecture of our proposed hybrid ranker. It consists of two separate modules that can independently compute the representations of the queries and information elements. Given a query $Q = t_{1 \cdots |Q|}$ where token $t_i \in V$ for vocabulary V, and a candidate information element $C = t_{1 \cdots |C|}$ of the same vocabulary, Our ranker first obtains lexical(l) and semantic(d) representations of the query and the candidate information as l_Q and d_Q, l_C and d_C, respectively following the process described in the next subsections. Then the relevance score of the information is computed by the linear interpolation of their semantic and lexical matching:

$$r(Q, C) = \alpha \times f(d_Q, d_C) + (1 - \alpha) \times f(l_Q, l_C) \quad (1)$$

Where $f(Q, C) = Q \cdot C$ and $\alpha \in (0, 1)$ is a hyperparameter indicating importance given to the semantic match.

4.2. Representation Learning

The representation learning procedure for query and information has independent yet similar pipelines as shown in Figure 2. The query-encoder is a pre-trained masked language model (MLM) such as BERT (Devlin et al., 2019) and it maps the query token sequence to their contextual embeddings $\boldsymbol{H_Q} \in \mathbb{R}^{|Q| \times h}$ (h: hidden size) and also outputs a summarised representation of entire query in the form of `[CLS]` token embedding $h_{CLS} \in \mathbb{R}^h$. While the sequence encodings can also be pooled to obtain the summarized vector, it requires additional computation. Instead, we use the pre-trained h_{CLS} as the query's dense semantic representation: $d_Q = h_{CLS}$.

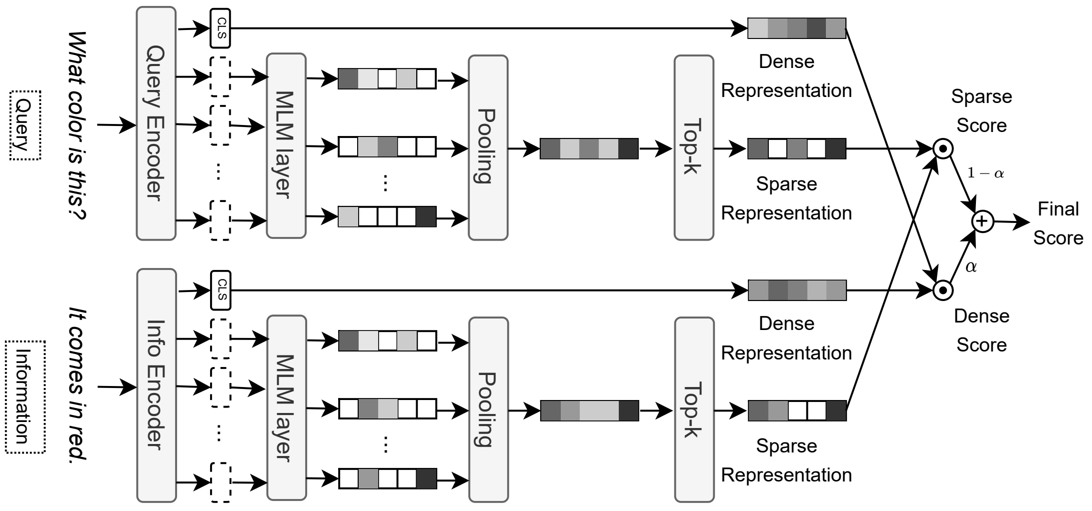

Figure 2: The proposed hybrid information ranker.

We build on the SPLADE (Formal et al., 2021) and SparseEmbed (Kong et al., 2023) methods to compute the lexical representation. In these methods, and as illustrated in Figure 2, the sequence encodings $\boldsymbol{H_Q}$ are fed to the BERT's pre-trained MLM head which maps them to MLM logits, $\boldsymbol{M_Q} \in \mathbb{R}^{|Q|\times|V|}$. Logit value $m_{i,j}$ in $\boldsymbol{M_Q}$ can be considered as an importance indicator of the vocabulary term $v_j \in V$ for the query token $t_i \in Q$. ReLU is applied to the raw logit values to ensure positivity and is followed by a log operation to reduce the dominance of fewer terms. Then the resultant logits are aggregated (using max-pooling or summation) along query token sequences to obtain the combined importance w_j of a term $v_j \in V$ using the following formula:

$$w_j = \max_{i=1\ldots|Q|} log(1 + ReLU(m_{i,j})) \qquad (2)$$

We collect the aggregated importance over the lexical terms, $W = w_{1\ldots|V|}$ through the max pooling layer. To reduce computational complexity during score calculation, we enforce sparsity in W by retaining only the top-k weights in it and zeroing out the rests as shown in Figure 2. This leaves us with an expansion-aware sparse lexical representation $l_Q = W$ of the query. Following a similar approach for the candidate information element, we obtain its dense semantic representation d_C and its sparse lexical representation l_C.

4.3. Loss Function

For the training of the hybrid model, we combine ranking loss due to both semantic and lexical representation. Given a dataset $S_{i\cdots|S|}$, where a training instance S_i comprises a query Q_i, a piece of positive information C_i^+ and b negative candidates $(C_{i,1}^-, C_{i,2}^-, \ldots, C_{i,b}^-)$, our model is trained to minimize the following contrastive loss for each kind of representation:

$$\mathcal{L}_{rank} = -\log \frac{e^{f(Q_i,C_i^+)/\tau}}{e^{f(Q_i,C_i^+)/\tau} + \sum_{j=1}^{b} e^{f(Q_i,C_{i,j}^-)/\tau}} \qquad (3)$$

Here, τ is a temperature hyperparameter. To have an efficient ranking system in terms of computational complexity and memory footprint, it is beneficial to enforce sparsity in the high-dimensional (size: $|V|$) lexical representation. Following SPLADE (Formal et al., 2021), we also use FLOPS loss for this regularization:

$$\mathcal{L}_{reg}^{C} = \sum_{j\in V}\left(\frac{1}{N}\sum_{i=1}^{N} w_j^{(C_i)}\right)^2 \qquad (4)$$

where C_i is a candidate information element in a batch of size N and w_j is the importance weight of a vocabulary token computed from Equation 2. Collectively, the training procedure minimizes the following loss function:

$$\mathcal{L} = \mathcal{L}_{rank}^{d} + \mathcal{L}_{rank}^{l} + \lambda^{Q}\mathcal{L}_{reg}^{Q} + \lambda^{C}\mathcal{L}_{reg}^{C} \qquad (5)$$

where λ^Q and λ^C are hyperparameters to introduce higher sparsity in query than information for less scoring cost.

4.4. Generator

Given a query Q and n number of potential information elements $C_{1\cdots n}$, we aim to generate an answer A. To effectively combine multiple information elements for a query, we employ a fusion-in-decoder (Izacard and Grave, 2021) model for answer generation. It uses a pre-trained sequence-to-sequence network such as T5 (Raffel et al., 2020)

#	Model	MAP	R-Prec	MRR@5	NDCG	Hit Rate@5	P@1
a	**BM25** (Robertson and Walker, 1994)	0.435	0.388	0.622	0.658	0.796	0.510
b	**Cross Encoder** (Nogueira and Cho, 2019)	**0.604**^{acdefghi}	**0.540**^{acdefghi}	**0.795**^{acdefgh}	**0.780**^{acdefghi}	**0.930**^{acde}	**0.703**^{acdefgh}
c	**Independent Dense** (Karpukhin et al., 2020)	0.552^{ade}	0.488^{ae}	0.761^{ade}	0.752^{ade}	0.918^{ade}	0.659^{ade}
d	**Late Interaction**(Khattab and Zaharia, 2020)	0.544^{ae}	0.481^{ae}	0.734^{ae}	0.741^{ae}	0.902^{ae}	0.618^{ae}
e	**Sparse Lexical** (Formal et al., 2021), k=128	0.505^{a}	0.449^{a}	0.694^{a}	0.713^{a}	0.873^{a}	0.572^{a}
f	**Hybrid, k=128**	0.563^{acde}	0.498^{acde}	0.770^{ade}	0.757^{acde}	0.924^{ade}	0.665^{ade}
g	**Hybrid, k=256**	0.572^{acdef}	0.505^{acdef}	0.780^{acdef}	0.763^{acdef}	0.925^{ade}	0.679^{acdef}
h	**Hybrid, k=512**	0.573^{acdef}	0.507^{acdef}	0.782^{acdef}	0.764^{acdef}	0.924^{ade}	0.679^{acdef}
i	**Hybrid, k=512, source-aware**	0.575^{acdef}	0.508^{acdef}	0.792^{acdefgh}	0.766^{acdef}	0.927^{acde}	0.697^{acdefgh}

Table 3: The overall effectiveness of the experimented rankers on the hetPQA (Shen et al., 2022) dataset. The best results are highlighted in boldface. Our hybrid model scores are obtained with $\alpha = 0.5$. Superscripts denote significant differences in both Fisher's randomization test and paired Student's t-test with $p \leq 0.05$.

that first encodes pairs of question and information $< (Q, C_1), (Q, C_2), \cdots, (Q, C_n) >$ independently and then joins the resultant representations in decoder before performing attention. Finally, we use greedy decoding to generate a natural language answer. As this method processes candidate information elements independently, it allows the aggregation of the elements at a relatively lower latency.

5. Experimental Environment

We use BERT-base-uncased (Devlin et al., 2019) (110M parameters) and T5-base (Raffel et al., 2020) (220M parameters) provided by Huggingface (Wolf et al., 2020) as the core model for evidence ranking and answer generation respectively. We set the following hyperparameters to the relevant models: {Max token length (each of question, evidence, answer): 128, Warm-up steps: 200, Batch size: 8, Gradient Accumulation Steps: 8, Learning rate: $1e-5$, λ^Q: $3e-4$, λ^C: $1e-4$}. The evidence rankers and generator are trained for 1,500 and 1000 steps respectively and the best checkpoints are considered for evaluation. All the experiments were conducted using a 5-core CPU node at 2.40 GHz, equipped with a single NVIDIA Tesla P100 16GB GPU core and 25 GB of memory. For preprocessing and evaluation, we use NLTK (Bird et al., 2009), calflops (xiaoju ye, 2023), and ranx (Bassani, 2022). Our baseline methods are listed in the first row of Table 3. We use Okapi BM25 implementation from rank_25 (Brown, 2020). For cross-encoder, independent dense encoders, and late-interaction method, we follow the implementation as described in §2. The only difference between the sparse-lexical method and our hybrid model is that the former does not incorporate semantic matching in computing the loss and the score. For fairness of comparison, none of our dual-encoders use any additional projection layer on top of BERT's layer and for ranking, we sorted all the candidate information based on Equation 1 instead of using any indexer.

6. Evaluation

In this section, we evaluate the performance of the two modules of our framework, viz., evidence ranking and answer generation.

6.1. Evidence ranking

We assess the impact of our proposed method along three dimensions: 1. ranking quality, 2. computational cost and memory footprint, and 3. interpretability. Table 3 lists the experiment results and provides a comparison of our proposed ranking method to the baselines specified in §5. Evaluation of all methods was conducted on the amended held-out test set and on the same environment as mentioned in §5. There are 2583 unique queries in the test set having at least one positive evidence and we consider only those queries for our evaluation.

Ranking Quality To report ranking quality, we utilize six commonly-used evaluation metrics- MAP: mean average precision, R-Prec: precision at the top-R retrieved information elements, MRR@5: mean reciprocal rank within top-5 candidates, NDCG: normalized discounted cumulative gain, Hit rate@5: fraction of queries with at least one positive evidence in top-5 ranked candidates and P@1: precision of the top-ranked evidence. As shown in Table 3, the hybrid approach outperformed all other methods except cross-encoder in all metrics. Although the hybrid model with $k = 128$ (top token count in lexical representation) bests the independent dense encoder model by a slim margin across the metrics, the difference in their effectiveness becomes statistically significant when more tokens ($k \geq 256$) are considered for lexical matching. The

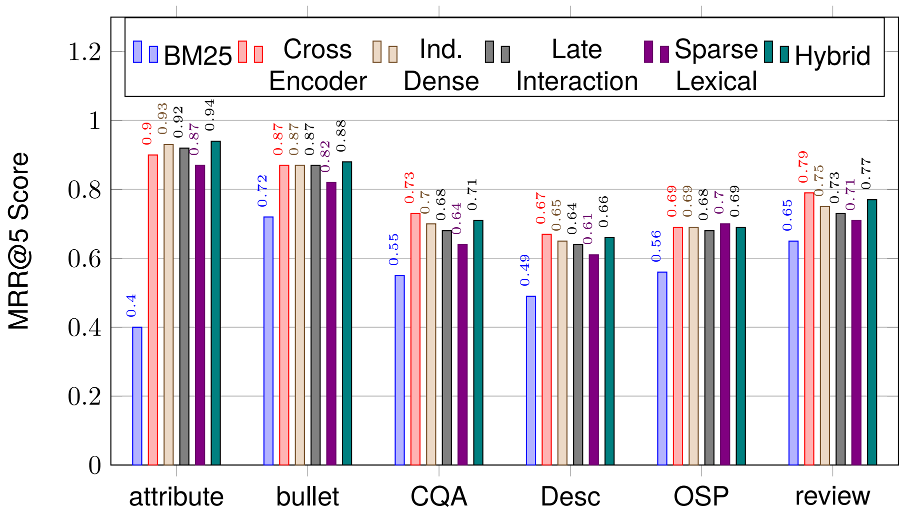

Figure 3: Ranking results of our hybrid ranker on heterogeneous evidence sources.

Resource Requirements							
Metric		BM25	Cross Encoder	Independent Dense	Late Interaction	Sparse Lexical	Hybrid
Params		-	109.48M	2x109.48M	2x109.48M	2x109.51M	2x109.51M
Inference FLOPs	Encoding	-	45.94G	22.36G	22.36G	28.51G	28.51G
	Interaction	-	-	$2h$	$2n^2 \cdot h + n$	$2k$	$2(h+k)$
Latency (ms)	Per Query	0.75	475.24	229.93	275.05	296.66	331.83
	Per Info.	0.007	4.2	2.04	2.43	2.63	2.93
Offline Storage (Per Evidence)		-	-	h	$n \cdot h$	$2k$	$h + 2k$

Table 4: Resource requirements of the experimented rankers on the hetPQA (Shen et al., 2022) dataset. Here dense representation size $h = 768$, max sequence length $n = 128$, Count of top tokens considered in lexical representation $k = 128$.

hit-rate@5 indicates the model positions at least one relevant piece of information among the top five in 92.7% of the queries.

Figure 3 illustrates a comparative performance of our proposed method with others across the six different sources of evidence. It shows that our hybrid model dominates existing methods in ranking evidence belonging to the same source. The contrasting score differences between BM25 and neural rankers in attribute and bullet sources not only show the struggle of the pure lexical method with less expressive data but also corroborate the advantage of semantic matching in handling heterogeneous data. In contrast to attribute or bullet evidence which stores clear and concise information, user-driven sources such as CQA and review come with inherent noise including misspellings, presumptive opinions, and so on. According to our manual inspection, these noises contributed to the models' relatively poor performance in these sources.

Resource Requirements Table 4 summarizes the resource requirements of our experimented ranking methods. All the methods employ the identical configuration of BERT (Devlin et al., 2019). Consequently, the parameter size listed in the first row is roughly proportional to that of a single BERT model except for the sparse-lexical and hybrid models where we have $2 \times 0.03M$ additional parameters for MLM layers. The second row provides the number of floating point operations (FLOPs) needed to be done in the inference stage which includes computation for encoding (measured in Giga-scale: 10^9) and interaction. Expectedly, the highly performing cross-encoder costs almost double the GFLOPs incurred by the independent dense encoder as the latter only performs query encoding in live and pre-computes the information representation offline. Late-interaction method, on the other hand, is subject to a quadratic interaction cost $(2n^2 \cdot h + n)$ due to its cross term-alignment. In contrast, our hybrid model outperforms all other two-tower rankers (Independent dense, Late-interaction, Sparse lexical) with a moderate 21% increase in encoding GFLOPs

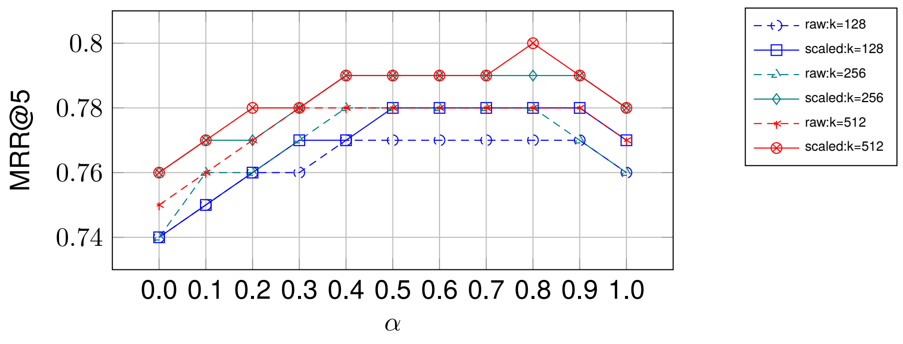

Figure 4: MRR@5 with regular (dashed) and source-scaled (solid) interaction scores at different semantic and lexical matching combinations.

and has a linear interaction cost as it only sums the product of the matching query tokens. For latency measurement, we consider the mean combined time elapsed for encoding, interaction, and score-sorting per query as well as per information. Each test set query has an average of 112 information elements. The inference latency is aligned with the inference FLOPs and the latency of our model is halfway between that of the independent dense model and cross encoder. In terms of offline storage required for each evidence representation, the hybrid approach demands space for a dense vector ($O(h)$) as well as key-value (key: vocabulary token index) pairs corresponding to non-zero elements ($O(k)$) of sparse lexical representation. This memory requirement ($O(h+k)$) is much smaller than that of the late-interaction method ($O(n \cdot h)$) as the latter stores all the token encodings.

6.2. Ablation Study

A comparison of evaluation results between our model and models using a subset of components reveals the contribution of additional components in our model. While results in all metrics show a similar trend, we use the standard MRR@5 for our ablation study. To begin with, our hybrid model is of identical architecture as in the sparse lexical model and differs from independent dense models only by the MLM layers. However, our model outperforms the sparse lexical model by 10.95% and the dense retriever by 0.7%-2.7% (for $128 \leq k \leq 512$) in MRR@5 (Table 3). This indicates the benefit of joint learning instead of maximizing only lexical or semantic matching independently.

While our semantic matching captures the underlying summarized meaning, explicit token matching compliments it by allowing us to interpret it. Figure 4 illustrates the effect of their contribution on MRR@5 by varying α in Equation 1 and changing sparsity i.e. the number of top-k tokens (represented by color) considered in sparse representation (see §4.2). The differences in area under curves indicate that a higher number of token considerations results in better ranking. On the other hand, the sub-optimal results with lexical-only ($\alpha = 0$) or semantic-only ($\alpha = 1$) matching and the consistently superior results with α in the range of $0.5-0.8$ further support our hybrid approach.

Furthermore, our analysis reveals that the distribution of scores across heterogeneous sources differs significantly and favors sources with high mean scores even if they obtain relatively low hit rates at top-5 positions. To counter this, we utilize the source-specific hit-rate@5 obtained from the regular ranking as prior confidence in those sources and multiply it with the combination of normalized scores obtained from Equation 1. The resulting ranking score, as shown by the solid lines in Figure 4, outperforms that of the regular ranking in dashed lines by 1%-3% across α and k values.

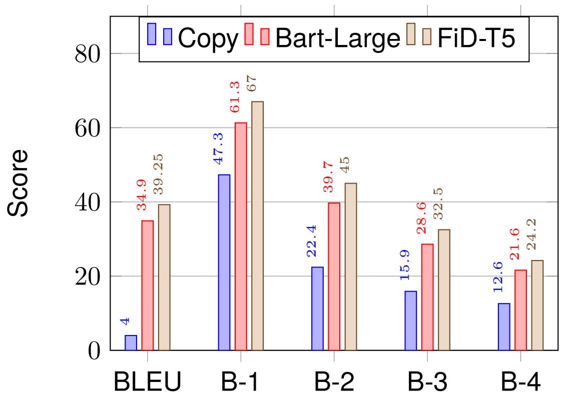

Figure 5: Results of answer generation.

6.3. Generation Quality

Figure 5 illustrates the answer quality generated by three approaches: 1. simply copying the top evidence as an answer, 2. Bart-Large (406M params) and 3. Fusion-in-Decoder with T5 (FiD-T5: 220M params). We utilize the results of the copy-based

Id	Source	Text	Expansion
#1 Desc	**Query**	how fast does the car go?	speed, time
	Evidence	maximum speed: 12 mph	fast, time, go
	Answer	the maximum speed is 12 mph.	
#2 Review	**Query**	how long do they stay lit?	time, last, light
	Evidence	the glow only lasts for on average of 30 minutes.	time, long, light
	Answer	they last under an hour.	
#3 CQA	**Query**	how do you hook it up to a television?	tv, power, plug
	Evidence	you just plug it directly to your tv.	power
	Answer	plug it into your television.	
#4 attribute	**Query**	how tall is the castle??	height, size
	Evidence	item dimensions width: $15.75''$, length: $30.5''$, height:$23''$	tall, size
	Answer	the castle is 23 inches tall.	

Table 5: Sample evidence prediction and answer generation.

approach and Bart-Large model from (Shen et al., 2022). Despite having a smaller number of parameters, the responses generated by FiD-T5 result in a higher BLEU score than that of other approaches. Examples of sample answer generation can be found in Table 5.

6.4. Interpretability Analysis - Examples and Discussion

A desired quality of a model is to have a simple and human-understandable mechanism to explain its decision-making process. Expanded tokens selected by our model's lexical representations can be interpreted as visualizable faces of underlying thoughts captured in jointly learned semantic representation. Further, the dot product of a matched token importance can be considered as its alignment strength. Table 5 illustrates this idea by highlighting matching tokens of query and predicted evidence. The importance of a token is depicted by its highlighting intensity. The examples demonstrate that our model can match relevant tokens through expansion even if they are not present in the original text. More interestingly, the matching expansion (e.g. *time* in ex#1, *light* in ex#2, *power* in ex#3 and *size* in ex#4) reveals the shared implicit impression that connects the query and the evidence.

There are a few shortcomings to the model which we leave as future work. First, it treats different forms (e.g. *lasts* and *lasting*) of a root token (e.g. *last*) as separate tokens causing redundant expansion. It can be avoided by merging them with their normalized value. Second, although we reduce the memory footprint of sparse lexical representation by keeping only token index-value pairs, further analysis is required to check its compatibility and efficiency with an indexer such as FAISS (Johnson et al., 2019). Without using such an indexer, despite having lower FLOPs, the ranking latency may rise dramatically if we compute token interaction in a loop. Furthermore, differential studies on domain-specific signals such as rate of product sale, count of repeating questions, customers' feedback, and engagement can be measured to quantify the effectiveness of the generator as well as the retriever.

7. Conclusion

The study presents a hybrid information ranker that ranks information for a query by comparing their jointly learned dense semantic representations and sparse lexical representations. Our evaluation found that our approach outperformed widely popular sparse or dense retrievers while incurring only a linear cost for both computation and offline storage. Also, our expansion-enhanced lexical matching demonstrates signs of interpretability. In the future, we plan to extend the framework to an end-to-end system with extensive evaluation using a larger dataset.

8. Bibliographical References

Yang Bai, Xiaoguang Li, Gang Wang, Chaoliang Zhang, Lifeng Shang, Jun Xu, Zhaowei Wang, Fangshan Wang, and Qun Liu. 2020. Sparterm: Learning term-based sparse representation for fast text retrieval. *arXiv preprint arXiv:2010.00768*.

Elias Bassani. 2022. ranx: A blazing-fast python library for ranking evaluation and comparison. In *ECIR (2)*, volume 13186 of *Lecture Notes in Computer Science*, pages 259–264. Springer.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: an-*

alyzing text with the natural language toolkit. " O'Reilly Media, Inc.".

Dorian Brown. 2020. Rank-BM25: A Collection of BM25 Algorithms in Python.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. Splade: Sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 2288–2292, New York, NY, USA. Association for Computing Machinery.

Luyu Gao, Zhuyun Dai, Tongfei Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan. 2021. Complement lexical retrieval model with semantic residual embeddings. In *Advances in Information Retrieval*, pages 146–160, Cham. Springer International Publishing.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 39–48, New York, NY, USA. Association for Computing Machinery.

Weize Kong, Jeffrey M. Dudek, Cheng Li, Mingyang Zhang, and Michael Bendersky. 2023. Sparseembed: Learning sparse lexical representations with contextual embeddings for retrieval. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 2399–2403, New York, NY, USA. Association for Computing Machinery.

Bing Liu and Ian Lane. 2016. Attention-Based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling. In *Proc. Interspeech 2016*, pages 685–689.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*, 9:329–345.

Ji Ma, Ivan Korotkov, Yinfei Yang, Keith B. Hall, and Ryan T. McDonald. 2021. Zero-shot neural passage retrieval via domain-targeted synthetic question generation. In *Conference of the European Chapter of the Association for Computational Linguistics*.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *ArXiv*, abs/1901.04085.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Stephen E. Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Dublin, Ireland, 3-6 July 1994 (Special Issue of the SIGIR Forum)*, pages 232–241. ACM/Springer.

Xiaoyu Shen, Gianni Barlacchi, Marco Del Tredici, Weiwei Cheng, Bill Byrne, and Adrià Gispert. 2022. Product answer generation from heterogeneous sources: A new benchmark and best practices. In *Proceedings of the Fifth Workshop on e-Commerce and NLP (ECNLP 5)*, pages 99–110, Dublin, Ireland. Association for Computational Linguistics.

Karen Sparck Jones. 1988. *A Statistical Interpretation of Term Specificity and Its Application in Retrieval*, page 132–142. Taylor Graham Publishing, GBR.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in NeurIPS*, pages 5998–6008.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

xiaoju ye. 2023. calflops: a flops and params calculate tool for neural networks in pytorch framework.

Hamed Zamani, Mostafa Dehghani, W. Bruce Croft, Erik Learned-Miller, and Jaap Kamps. 2018. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 497–506, New York, NY, USA. Association for Computing Machinery.