

A DATASET COLLECTION

A.1 DATASET ANNOTATION

Filtering The original HybridQA dataset contains over 72k questions paired with 13k hyperlinked tables. We adopt two filtering heuristics to make the decontextualization easier. First, we filter out tables without enough meta-information or containing too much non-textual information². Second, we filter out overly-long questions, i.e., questions longer than 30 words. These two filtering heuristics result in a cleaner subset of 46k questions paired with 9k in-domain tables.

Quality Control During annotation, we conduct strict manual quality evaluation over the decontextualized questions, with the following criteria: 1) the annotated question retains the same semantics and answer as before, 2) the annotated question still requires multi-hop reasoning over both table and passages, and 3) the annotated question is concise and fluent. The manual quality checking was performed over batches distributed to the same annotator. Each batch consists of six questions, one of which will be sampled to decide the acceptance/rejection of the whole batch. The overall acceptance rate for the crowd-sourcing job is 71%, and a rejected job was re-distributed until it was accepted.

A.2 DATASET EXAMPLES

We demonstrate more examples in Figure 7, which includes more diverse inference chains, like table \rightarrow text; text \rightarrow table \rightarrow text; text + text \rightarrow comparative \rightarrow table. Our model is able to perform these reasoning types quite well by jointly matching a query against a fused table-text block.

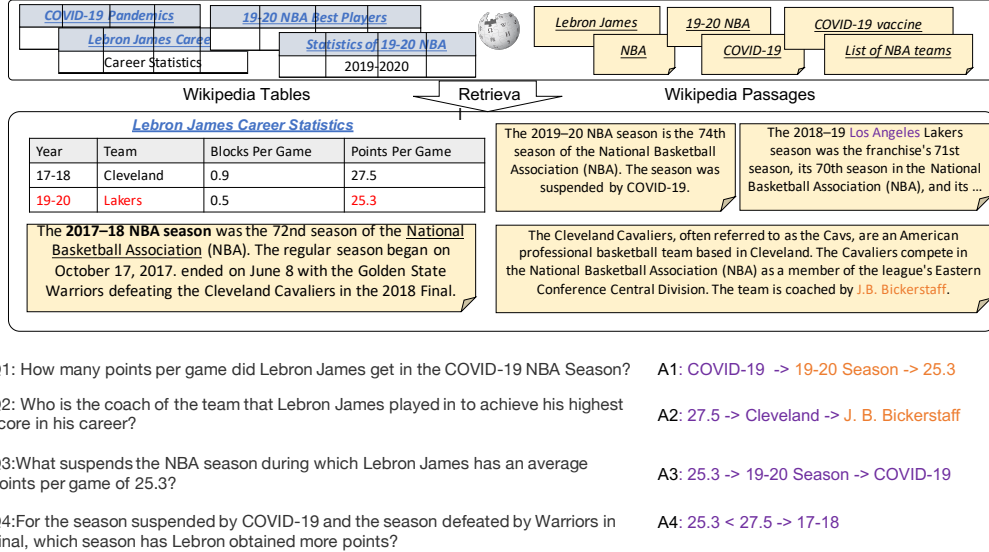


Figure 7: More examples from OTT-QA

A.3 QUESTION TYPES

We randomly sampled 100 questions from the dataset to manually analyze the kinds of inference chains seen in OTT-QA and divide the major types into the following categories:

1. Single hop questions (13%) require reading one table or one passage to answer.
2. Two hop questions (57%) require reading one passage and one table to answer. These can be subclassified as ‘table bridge’ \rightarrow ‘answer text’³ or ‘text bridge’ \rightarrow ‘answer table’.

²Like longitude, latitude, mathematical formulae, etc

³I.e., a table forms a bridge between a question and a textual passage, which is read in the first hop, and the answer is extracted from the text.

3. Multi-hop questions (30%) require reading two passages and one table to answer. These mainly following the reasoning chain of ‘text bridge’ \rightarrow ‘table bridge’ \rightarrow ‘answer text’.
4. Questions with multiple reasoning paths: Due to information redundancy in Wikipedia, similar information can appear in both tables and text. We find that 9% of questions are answerable by reading one text passage, 18% of questions are answerable by reading two text passages and 4% of questions are answerable by reading two tables.

B MODEL DETAILS

B.1 RETRIEVAL BLOCK REPRESENTATION

The table decomposition is visualized in Figure 8. The title/section title are prefixed to the table segment. We add the row position token ‘1st’ and a max/min special token over the column to infuse global table information into the segmented unit. The column embedding is added as another vector to the representation. The table segment representation is relatively small and easy to deal with in the following reader model. After the table-passage alignment, we group the highly related

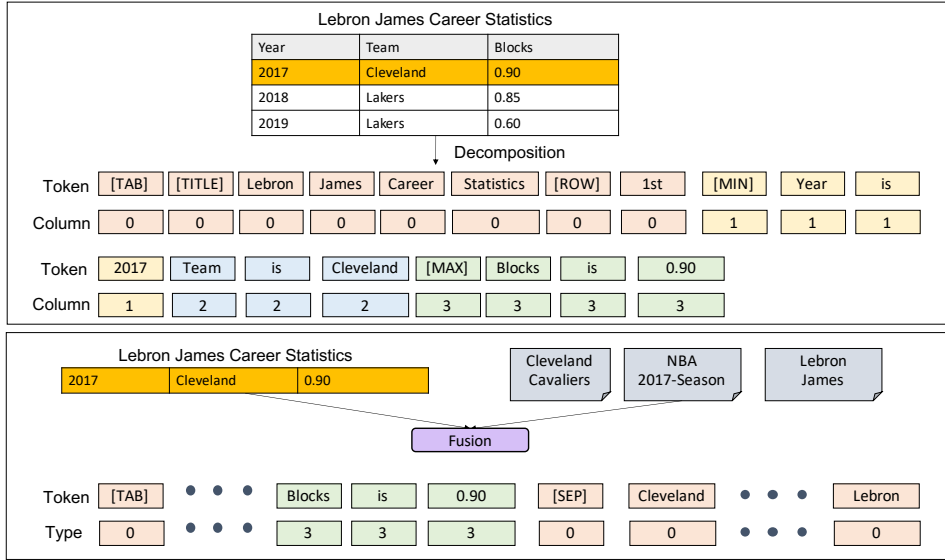


Figure 8: The decomposition of the original table into segments.

units together and represent them as the lower part demonstrated in Figure 8. We add [SEP] tokens to separate different passages and set their type id to 0. Such a flattened representation for fused block b_F will be used throughout our experiments for both sparse/dense retriever and ETC reader.

B.2 ITERATIVE RETRIEVER

Iterative retrieval has been used in recent graph-based multi-hop retrieval models to gradually retrieve documents to find the correct supporting evidence. Specifically, the retriever conditions the i -th round retrieval on the previous round of retrieval results.

Sparse Retriever The sparse retriever uses uni-gram lexical feature to compute the BM-25 score between the $q, \dots, b_{1..j-1}$ over $b_j \in \mathbb{B}$ to get the top candidates. Here we describe a two-step retrieval procedure, called here the LxM procedure. In the first step, the model calculates the BM25 score between question over all the candidates in \mathbb{B} to select top L/2 table segments b_T , and L/2 passages b_P . In the second step, the question is concatenated with the retrieved table segment to form L/2 new queries $[q; b_T]$ which are used to retrieve LM/2 passages from \mathbb{B} . The question is also concatenated with the retrieved passage titles to form another K/2 queries $[q; b_P]$ to retrieve LM/2 table segments. The retrieval procedure results in at most LM+L unique blocks. Each unique block aggregates its

score from two rounds, denoted as $f(q, b)$, which is used to rank the top-K candidates for the next step. We truncate the top-K candidate by thresholding their combined length.

Dual-Encoder Retriever The dual encoder uses a BERT-based encoder to compress each question, table segment, and passage into a fixed-length vector and then computes the dot product between fixed vectors to obtain the highest scored candidates from pool \mathbb{B} . However, since the dataset does not provide an explicit supervision signal for the iterative retrieval, we heuristically synthesize some noisy retrieval chains using lexical matching. The retrieval inference chain is depicted as $b_1 \rightarrow b_2 \rightarrow b_K$, which is used to train the model $f(b_k|q, b_{1..k-1})$ in a supervised manner. At inference time, the dual encoder retriever will encode a query q into a fixed vector and retrieve the first L blocks from \mathbb{B} . The blocks are appended to query q to form L new queries $[q; b_i]$, which is re-encoded and search for LM new neighbors. We experiment with a maximum of 3-step retrieval of $L \times M \times N$ to obtain a maximum of $L+L \times M+LMN$ unique blocks. Similarly, each unique block aggregates its score from different rounds to select the top-K candidates for the next step.

B.3 SPARSE FUSED RETRIEVER

The sparse fused retriever uses the uni-gram lexical feature to compute the BM-25 score between q over $b_F \in \mathbb{B}_F$. The uni-gram feature of b_F is based on the representation depicted in subsection B.1. Note that this BM25 feature will be much more abundant than the BM25 feature in iterative sparse retriever because it encloses more uni-grams. Instead of doing multiple rounds of retrieval, the fused retrieval once retrieve once over the candidate pool and treat all the units inside the block as the same retrieval score. Finally, We truncate the top-K candidate by thresholding their combined length.

B.4 QUERY AUGMENTATION

The query augmentation procedure is depicted in Figure 9.

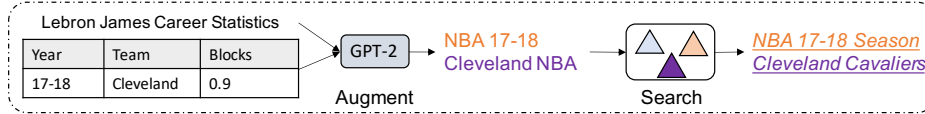


Figure 9: Fusion: 1) GPT-2 query augmentation, 2) nearest neighbor search over passages.

B.5 DENSE RETRIEVAL/IN-BATCH NEGATIVE

Recently, different dense-retrieval methods (Lee et al., 2019; Guu et al., 2020; Karpukhin et al., 2020) based on dual-encoders (Bromley et al., 1994) have been shown to surpass traditional sparse retrieval in open-QA models. The query and the passage are both encoded using a Transformer, which produces a vector for every token. As in (Devlin et al., 2019), the vector corresponding to the first token, $[CLS]$, which is used as a “pooled” representation of the sequence (denoted $BERT_{CLS}(q)$). The dense retrieval function can be represented as the dot product between $BERT_{CLS}(q)$ and $BERT_{CLS}(p)$ for each document in the text collection, much like TF-IDF (Chen et al., 2017) and BM25 (Robertson & Zaragoza, 2009) on some Open QA datasets. To train the dual-encoder, the in-batch negative trick (Yih et al., 2011; Karpukhin et al., 2020) plays an important role, which uses B training instances in each batch and views the other $B-1$ instances inside the batch as the negatives. In this way, the model reuses computation and effectively trains on B^2 question/document pairs in each batch.

C PERFORMANCE ANALYSIS

C.1 QUESTION TYPE BREAKDOWN PERFORMANCE

We measure our best model’s performance (dense fusion retriever + cross-block reader) and baseline model (dense Iterative-Retriever + single-block reader) on different question types (subsection A.3) to show the breakdown statistics in Figure 10 and Figure 11. As we can observe, the gap between

our model vs. baseline in 1-hop question is less significant as 2-hop and 3-hop questions. The iterative retriever’s performance is sensitive to the number of hops in the question, which is the largely due to the error propagation in the beam search stage. If the retriever fails to include the golden block in the earlier stage beam, the retrieval in later stage cannot recover from such failure. In contrast, our fusion retriever can group the related information prior to retrieval to retrieve all the blocks at once, which makes the model less prone to the error propagation issue. Another reason is due to the cross-block reader, which can reason over different blocks in the latent space, such implicit reasoning can also decrease the error propagation issue. To sum up, our model is more powerful to deal with complex multi-hop open questions with much less performance drop.

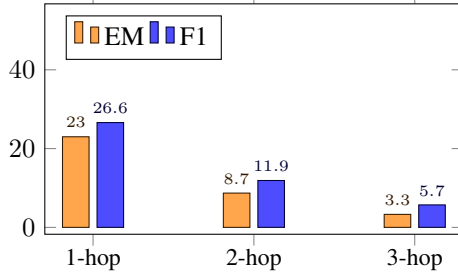


Figure 10: Breakdown for iterative retriever + sing-block reader.

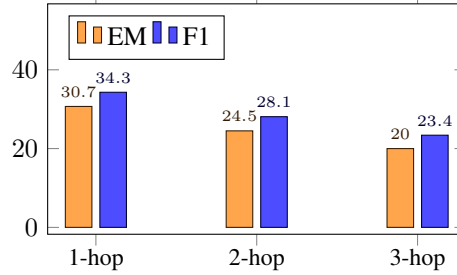


Figure 11: Breakdown for fusion retriever + cross-block reader.

C.2 RETRIEVER ERROR ANALYSIS

We conduct error analysis to see what are the major issues with the retriever and conclude the following types in Figure 12. The major issues causing the system to retrieve unrelated evidence are low lexical overlap, fusion errors, numerical reasoning and distracting passages or tables. In the

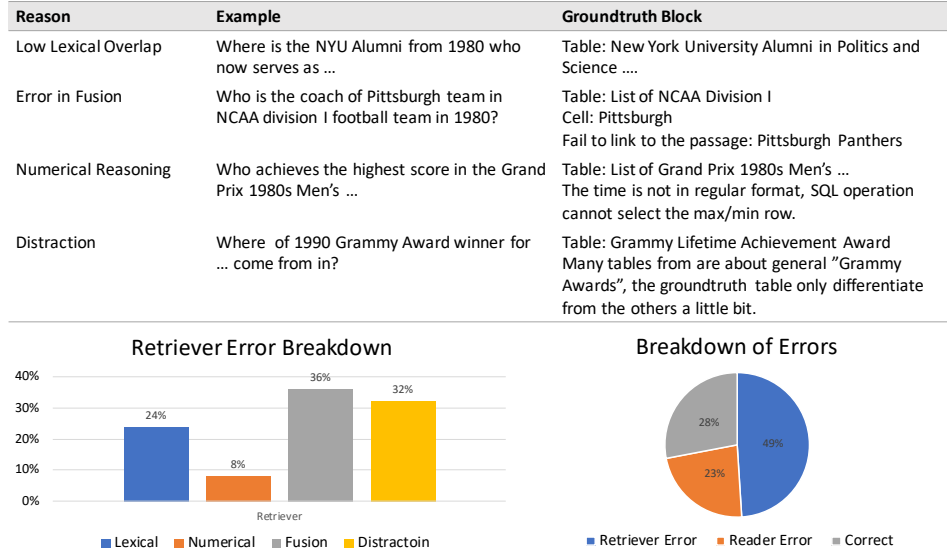


Figure 12: The main error types in the retriever.

Low-Lexical-Overlap case, the errors are mainly coming from the abbreviation, rephrasing of the table metadata, for example, ‘New York University’ is shortened as ‘NYU’, etc. In the Fusion-Error case, the issue is mainly because the entity-linking model fails to fuse all the hyperlinked passages, the error (F1=50%) is quantitatively reflected in the entity-linker-performance figure. Numerical-Reasoning error is mainly related to the failure to find max/min/earliest/latest row in the table. The

distraction error is mainly caused by some distracting passages or tables having very similar information. We sample 50 error samples from the dev-set and attribute their errors to the above categories. As shown in the left part, we found that the numerical reasoning error is not as severe as the other three types because the proportion of questions requiring it is relatively small. Besides the low-lexical overlap error, which is general across other open QA datasets like NQ and HoptpotQA, we found the fusion and distraction errors quite specific in our dataset.

- (Fusion) Questions which ask about tables which are linked to too many linked passages. For example, a question over table “Team Record” in https://en.wikipedia.org/wiki/Sevens_Grand_Prix_Series is hard because some table rows associate with over 10 passages, it’s hard to link them and fuse all of them into a fused block.
- (Distraction) Questions which ask about topics which are contained by too many similar tables, it’s hard to differentiate the true one. For example, there are over ten tables in https://en.wikipedia.org/wiki/List_of_RMIT_University_people, these similar tables can easily distract the attention of the retriever to select the wrong one from the same page.

From our quantitative results, we can attribute the errors to retriever and reader, among all the examples, 49% of examples cannot find the correct supporting block. For the rest 51% examples with correct block retrieved, the reader fails to select the correct span for 23% of them.

C.3 LENGTH SENSITIVITY ANALYSIS OF RETRIEVAL/READER

We perform sensitivity analysis for both retriever and reader in Figure 13. We gradually increase

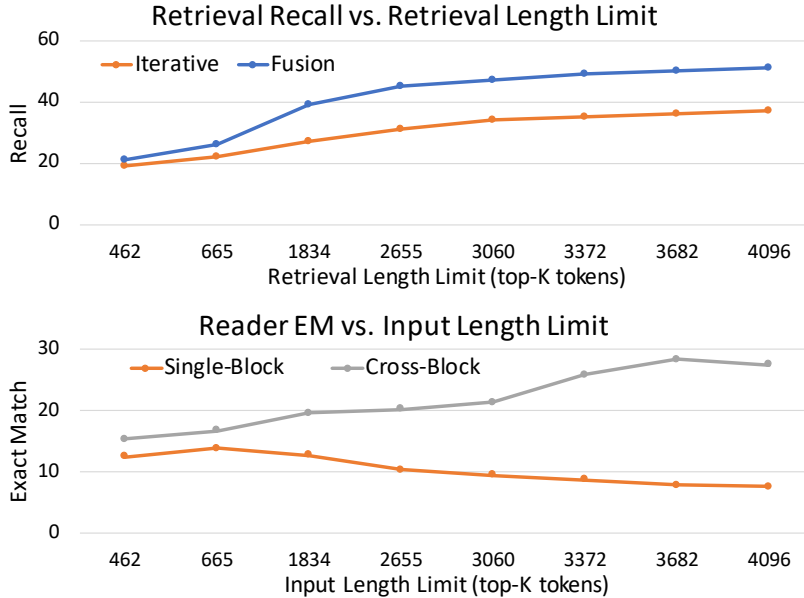


Figure 13: Analyzing retriever performance.

the length limit of retrieved evidence from 400 to 4096 to first visualize its impact on the sparse iterative and dense fusion retriever. For both fusion and iterative retriever, we can observe that both of their recall@K significantly improves as the length limit increases. With a low budget of token limit, their performance gap is smaller because its performance is dominated by the single-hop questions in the dataset. As the length limit increases, the improvement for fusion retriever is steeper than iterative retriever because the contextualized fusion block becomes easier to retrieve than standalone table segment or passage.

We also visualize the input length’s impact on the single-block The performance of single vs cross-block reader. With a low budget of token limit, both single-block and cross-block readers are comparable. However, as the limit increases to 4000, the cross-block reader can digest long input with

its sparse attention mechanism to achieve better scores, while the single-block reader needs to truncate the information to read independently, which leads to a even lower EM score due to introduced noise. This observation reveals the importance of modeling cross-attention between different retrieved evidence units to reach a consistent answer. In single-block reader, dealing with different blocks independently can lead to suboptimal prediction in our dataset.

D CONNECTION TO EXISTING WORK

KB and Text The problem combining structured and unstructured data has been studied in question answering. The previous approaches are mainly divided into two categories: 1) FusionNet and PullNet (Sun et al., 2018; 2019) simulate a KB-incomplete setting by masking out some triples from a knowledge graph and use textual information to complete the masked KB triples; these experiments are conducted on KB-based QA datasets. 2) DrKIT (Dhingra et al., 2019) and Knowledge-Guided Retrieval (Min et al., 2019) propose to use entity mentions and relations to guide the retrieval from the web. However, the KB is mainly used as an assisting tool, rather than a necessary information source. In OTT-QA, the structured data is used as necessary information in a realistic setting. The two information forms are combined in a non-trivial way, which makes the problem much harder than the other structure-unstructured QA settings.

Entity Linking Our generative entity linker is related to knowledge-enhanced language understanding (Petroni et al., 2020), which proposes a seq2seq model to deal with different knowledge-intensive tasks like slot filling, entity linking, etc. There is a concurrent related work on auto-regressive entity linking (De Cao et al., 2020), which also demonstrates the advantages of using an autoregressive generation model for entity retrieval.