Explicit Question Decomposition for Multihop Question Answering Retrieval

Neha Nishikant

CMU-CS-22-149

December 2022

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Eric Nyberg, Chair
Graham Neubig

Submitted in partial fulfillment of the requirements for the degree of Master of Science.

Keywords: natural language processing, multihop question answering, information retrieval, dense retrieval

Abstract

Open domain question answering is a problem in NLP where a system answers learns to answer questions based on a large corpus of documents. Commonly, a retriever model first retrieves relevant documents and then a reader model extracts the correct answer. We specifically seek to improve the retrieval of ”multihop questions”, which are questions that can be decomposed into multiple subquestions, making them more complex and realistic. We explore if using the explicit question decompositions in the retriever model can improve results. We define a model MEX and MEX-oracle and perform experiments on a multihop QA dataset. Our results show that explicit decompositions are useful for sparse retrieval models, but not for dense retrieval.

Acknowledgments

I would like to thank Eric Nyberg for advising and guiding me this past year to become more knowledgeable and a better researcher overall, and Graham Neubig for helping me begin my research journey and introducing me to the world of Question answering.

I would also like to thank Shuyan Zhou and Kaixin Ma for their invaluable mentorship over the past few semesters.

Lastly, thank you to my family and friends for their unwavering support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Open domain question answering is a task for automated models to learn how to retrieve answers to questions from a large corpus of documents/passages. A popular paradigm is the retriever-reader model where the retriever selects relevant passages and the reader scans the passages to find the correct answer span in the text [1].

There is a lot of preexisting work about question answering for simple questions where the answer to the question lies in a single passage. [14]. However, the concept of multihop questions is a more accurate reflection of questions in the wild [10]. Multihop questions are more complex and are composed of multiple simple sub-questions where each sub-question has its own relevant passage and answer span. For example, "Did Aristotle use a laptop" can be decomposed into the following [3]:

- When did Aristotle live?
- When was the laptop invented?
- Is #2 before #1?

The answer to each of these sub-questions can be found in different passages in the corpus, which is what makes retrieval for this problem challenging.

In the era of the internet, open domain question answering is more relevant than ever. As the internet grows more saturated, it grows more important to find ways to continue to understand and synthesize it. [9] introduces the idea of using the web as a corpus for retrieval for question answering. For our purposes, we use a Wikipedia corpus which divides each document into multiple passages. This corpus is commonly indexed with fast indexers like FAISS or ElasticSearch [9]. Given this corpus, we can build systems to answer questions about.

Chapter 2

Related Work

2.1 ORQA

Before [5], an Open Retrieval Question Answering system (ORQA), retrieval was sparse, done via word frequency . Word vectors were sparse because each index represented the presence of its respective token in the passage, thus many indices were 0. However, [5] improved dense, learned methods drastically using a auxiliary task called inverse cloze task (ICT). Dense methods learn semantic word embeddings and all indices are used. Later, Dense Passage Retrieval (DPR) improved upon ORQA and removed the need for ICT via intelligent training using positive and negative passages [4].

2.2 MDR

One current state of the art model for multihop question answering is Multihop Dense Retrieval (MDR) introduced in [12]. It works similarly to DPR in that it trains an RoBERTa based encoder to encode passages and questions in a way that captures semantic meaning [4]; [6]. Then it uses a similarity metric to output the highest scoring passages for a given question. Specifically, MDR was evaluated on a popular QA benchmark/dataset HotpotQA [13]. The score of a passage is given by the inner product of the query and the passage.

What makes MDR different from DPR is the focus on multihop questions. Since answers to each sub-question may be in different passages, MDR iteratively performs this retrieval n times, where n is a hyperparameter that implicitly denotes the number of subquestions/hops. Note, we say "implicitly" because MDR does not actually rely on explicit question decompositions. Instead it relies on the fact that the question has some inherent but unknown multihop structure. At each hop, MDR reformulates and reencodes the query with added information. At the first hop, the query is just the main question. At each next hop, the passage retrieved from the previous hop is added to our query before encoding. These previously retrieved passages are called the *context*.

The learned encoder has a shared base with the passage encoder and is trained on gold standard passages for each subquestion.

$$P(p_t|q, p_1, ..., p_{t-1}) = \frac{\exp(\langle \boldsymbol{p}_t, \boldsymbol{q}_t \rangle)}{\sum_{p \in \mathcal{C}} \exp(\langle \boldsymbol{p}, \boldsymbol{q}_t \rangle)}, \text{ where } \boldsymbol{q}_t = g(q, p_1, ..., p_{t-1}) \text{ and } \boldsymbol{p}_t = h(p_t).$$

Figure 2.1: Probability of passage p_t being the correct passage at hop t under Multihop Dense Retrieval given the initial question q and the passages retrieved at previous timesteps up to $t-1$. This is the probability that MDR tries to maximize. Note that q_t, the query at timestep t is the question concatenated with the $t-1$ previously retrieved passages.

Figure 2.1 below shows how the authors modeled the probability for choosing passage pt at timestep t given the question q and the previously retrieved passages p_1 through p_{t-1}.

2.3 Decompositions

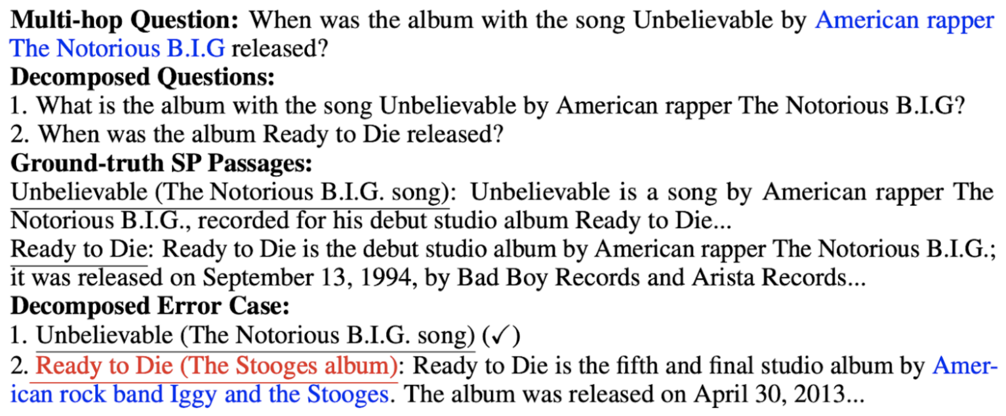
One common idea in the QA community that MDR does not leverage is explicit question decomposition. The list of subquestions for a given question is called the question decomposition. Previous work has used knowledge bases for multihop open domain question answering, but knowledge bases require a lot of human design decisions and they can be hard to populate [2]; [7]. Thus, language models have have risen to state of the art.

[11] showed that using explicit decompositions helps when using TF-IDF as their retrieval method. Later, [12] hypothesized that this does not hold for dense methods. In particular, they found that for MDR, there are no performance improvements. The authors argue that the implicit iterative retrieval alone gets enough information required at each hop.

However, we hypothesize that the reasons for this result may be because of the configuration in which they use the question decompositions. They provide three examples of retrieval errors made only in the decomposed version of the system, not plain MDR. In these examples, the error can potentially be attributed to the information loss that happens from one decomposed question to the next. In particular, consider the provided example in figure 2.2.

The red is the error and the blue is information that would be useful in finding the correct passage. As we can see, there is some information loss between the first and second decomposed question. In the second decomposed question, it is unclear that “Ready to Die” refers to The Notorious B.I.G album, not The Stooges album. [12] also write that these errors could “potentially be addressed by a different style of decomposition” and “access to the full information in the original question or previous hop results”.

Another possible explanation for the observed lack of improvement between plain MDR and using question decomposition is that the “decomposition is restricted to a small vocabulary derived almost entirely from the original question” [3]. This explains why using question decompositions did not add much information as compared to just using the original question.



Multi-hop Question: When was the album with the song Unbelievable by American rapper The Notorious B.I.G released?
Decomposed Questions:
1. What is the album with the song Unbelievable by American rapper The Notorious B.I.G?
2. When was the album Ready to Die released?

Ground-truth SP Passages:
Unbelievable (The Notorious B.I.G. song): Unbelievable is a song by American rapper The Notorious B.I.G., recorded for his debut studio album Ready to Die...
Ready to Die: Ready to Die is the debut studio album by American rapper The Notorious B.I.G.; it was released on September 13, 1994, by Bad Boy Records and Arista Records...
Decomposed Error Case:
1. Unbelievable (The Notorious B.I.G. song) (✓)
2. Ready to Die (The Stooges album): Ready to Die is the fifth and final studio album by American rock band Iggy and the Stooges. The album was released on April 30, 2013...

Figure 2.2: From the appendix of MDR, an example query from the authors of [12] where using explicit decomposition resulted in incorrect retrieval. The "Decomposed Error Case" shows that the retrieved passage for the second subquestion is wrong. The model retrieved information about an different album with the same name.

While their results are impressive, We hypothesize that the power of explicit question decompositions has been underutilized. Chiefly, we want to explore if using explicit question decompositions can help performance for dense retrieval methods for open domain question answering.

Chapter 3

Method

3.1 Dataset

To test this hypothesis, we analyze performance of a decomposition-based retriever on the StrategyQA dataset introduced by [3] which is a dataset of multihop questions with gold annotated passages, decompositions, and answers for each question.

The dataset is constructed via a few tasks: Creative Question Writing, Strategy Question Decomposition, and Evidence Matching [3]. These tasks help minimize bias in crowd sourcing. During Creating Question Writing, workers are given as much freedom as possible for question topic and wording and are primed by Wikipedia terms to help ensure feasiability of the question. During Strategy Question Decomposition, workers have an auxialiary task of predicting the title of a Wikipedia article that may answer their subquestion, to further ensure feasiability. Finally, there is Evidence Matching where workers find passages to answer subquestions.

We work with the dev split which has 229 questions. Each question has 3 annotators, each of which denote a subquestion as a ”retrieval” step with its corresponding gold-standard passages, or an ”operation step” which means the answer can’t be retrieved and requires a logical operation. For example for the decomposition of ”Did Aristotle use a Laptop?”,

- When did Aristotle live?
- When was the laptop invented?
- Is #2 before #1?

the 3rd subquestion is an operation step whose answer cannot be found in a passage.

3.2 Baseline Models

The baseline model for StrategyQA is coined RoBERTa*, a version of RoBERTa that was fine-tuned on DROPQ, 20Q, and BOOLQ [3]; [6]. On top of those baseline, [3] introduced RoBERTa* IR-Q which follows the retrieval-reader paradigm. The retrieval model was BM25, a sparse, frequency-based, retrieval model defined in [8].

There are a couple reasons why improving the retrieval for RoBERTa* IR-Q is a plausible way to increase the accuracy. These reasons are motivated by the results in Table 3.1 [3].

First, while RoBERTa* IR-Q shows no significant accuracy improvement over plain RoBERTa*, [3] also analyzed the performance analysis of RoBERTa* ORA-P, an oracle version of RoBERTa* IR-Q. Instead of using BM25 as the retrieval, the oracle model simply uses the gold standard passages. The accuracy of RoBERTa* ORA-P is significantly higher than that of RoBERTa*, indicating that better retrieval should improve the accuracy of RoBERTa* IR-Q [3].

Secondly, recent work has shown that dense, learned retrieval methods perform much better than sparse methods such as BM25 or TF-IDF [5]. The rationale here is the dense methods can link semantically similar words, even if they're not the same exact word, like "bad" and "villain" [4].

3.3 Multihop Explicit MEX

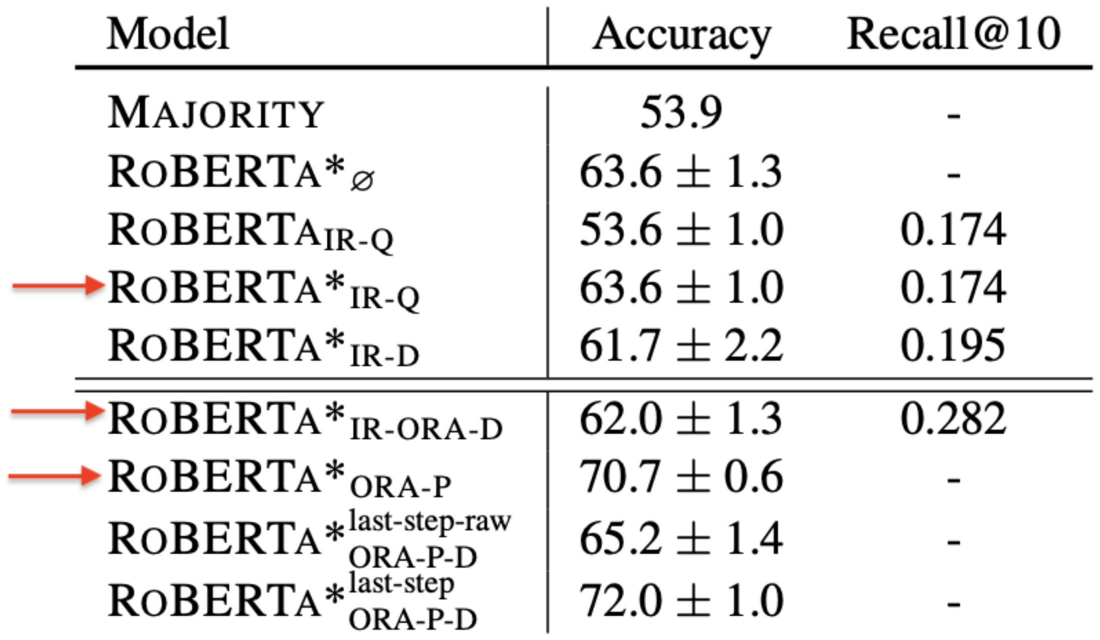
We propose a model, Multihop EXplicit (MEX), seen in Figure 3.1,extended from MDR in order to mitigate the information loss experienced when authors when they tried to use explicit question decomposition, as shown in Figure 2.2.

For MEX, we keep a few key ideas from MDR. In particular, we keep the sequential passage retrieval and the concatenation to keep information from hop to hop. However, instead of using the main question at each hop, we use the subquestion at each hop. Further, MDR assumes an equal number of subquestions for each question. In particular, in their implementation, they assume 2 hops. This assumption in integral to their code.

However, because we use the annotated explicit decomposition, the number of subquestions is variable. Thus MEX dynamically creates the queries at runtime. Specifically, the query for MEX at hop i is the i^{th} subquestion concatenated with the the $i-1$ previously retrieved passages. This query is encoded and then passed into the model.

We hypothesized that maintaining all the previous hops' retrieved passages would mitigate the aformentioned information loss. Consider the example in Figure 2.2. Instead of the second query being simply "When was the album Ready to Die released?", as it is in Figure 2.2, the MEX query would also include the passage retrieved at the previous hop. The gold-standard passage is "Unbelievable is a song by American rapper The Notorious B.I.G, recorded for his debut studio album Ready to Die...". Because the MEX query includes this passage, the model should hypothetically pick up on the information that "Ready to Die" is the The Notorious B.I.G album, not the The Stooges album.

Further, note that when the authors of MDR tried explicit question decomposition, they filled in the correct answers filled into the blanks in the subquestions. For example, for the decomposition provided for "Did Aristotle use a laptop?", we don't know the answers to subquestions 1 and 2 in advance, so they are represented as blanks in question 3 [3]. In order to use the explicit question decompositions, we need to resolve these blanks by answering questions 1 and 2.



Model	Accuracy	Recall@10
MAJORITY	53.9	-
RoBERTa*$_{\varnothing}$	63.6 ± 1.3	-
RoBERTa$_{\text{IR-Q}}$	53.6 ± 1.0	0.174
→ RoBERTa*$_{\text{IR-Q}}$	63.6 ± 1.0	0.174
RoBERTa*$_{\text{IR-D}}$	61.7 ± 2.2	0.195
→ RoBERTa*$_{\text{IR-ORA-D}}$	62.0 ± 1.3	0.282
→ RoBERTa*$_{\text{ORA-P}}$	70.7 ± 0.6	-
RoBERTa*$^{\text{last-step-raw}}_{\text{ORA-P-D}}$	65.2 ± 1.4	-
RoBERTa*$^{\text{last-step}}_{\text{ORA-P-D}}$	72.0 ± 1.0	-

Table 3.1: Results from the StrategyQA baseline models. We've specifically highlighted RoBERTa* IR-Q, since that model uses BM25 for retrieval whereas the other models don't use any retrieval, RoBERTa* ORA-P, since that is an oracle model that uses gold-standard passages instead of learned retrieval, and thus its results represent the best possible scenario, and RoBETa* IR-ORA-D, since this model uses gold explicit question decomposition. You may notice that the question answering accuracy for RoBERTa* IR-Q and RoBERTa* IR-ORA-D are comparable of that to RoBERTa* , which uses no retrieval method. While this is discouraging, we strongly believe that this is due to limitations of StrategyQA's reader model because even for oracle models like RoBERTa ORA-P, the accuracy it is quite low for a binary question since 50% accuracy can be achieved by randomization alone. In this thesis, we are mainly concerned with improving retrieval (measured by recall) as explained in section 3.2.

However, we hypothesize that the contexts appended to the query from previous hops can provide enough information to the model without explicitly filling in these blanks. We have a few reasons for making this design choice for MEX. Firstly, we don't have gold standard answers for each subquestion. Attempting to fill in these answers using our own annotations would add in human bias apart from the annotators in StrategyQA. Attempting to fill in these answers using a state of the art reader model could introduce error propagation. And further, leaving the blanks as blanks removes any supposed advantage that explicit decompositions might have from having access to gold standard answers at each hop.

Lastly, we limit MEX's retrieval to subquestions for which more than half the annotators marked it as a retrieval step as opposed to an operation step. For StrategyQA, there are 3 annotators, so we only retrieve for a subquestion if at least 2 of the annotators have marked it as a retrieval step.

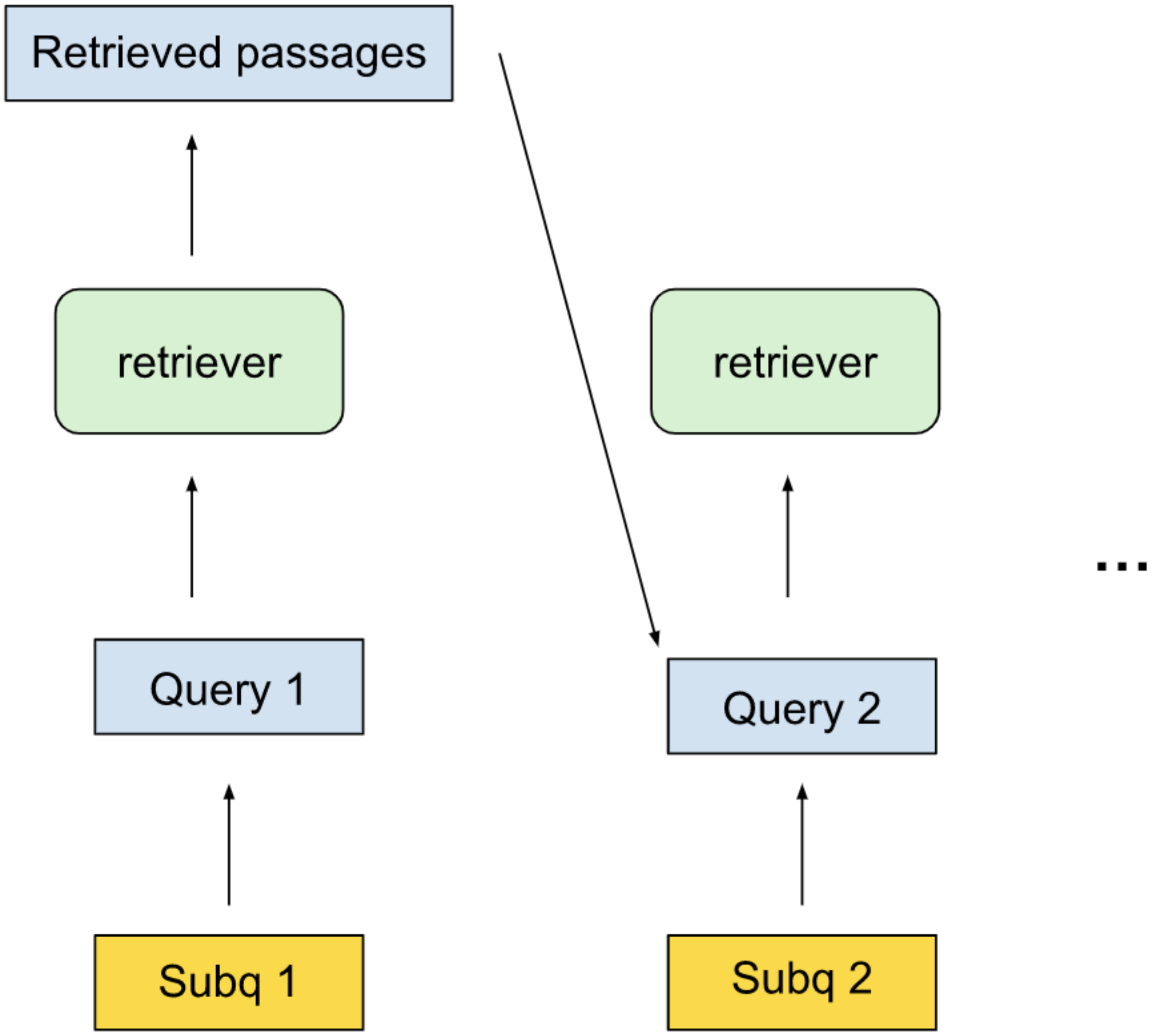


Figure 3.1: MEX model architecture

Chapter 4

Results

4.1 Evaluation

Our goal was to discover if explicit decompositions improved retrieval for StrategyQA. The metric we use to evaluate retrieval is recall.

We used four models

- **BM25**: This represents the retrieval for StrategyQA's baseline model RoBERTa* IR-Q.
- **BM25**$_{decomp}$: This is a respecification of retrieval for StrategyQA's baseline model RoBERTa* IR-ORA-D as shown in Figure 3.1. This model works by performing BM25 retrieval on each subquestion. The top retrieved passages for each decomposition are aggregated as the set of retrieved passages for the overall question. The main difference between BM25$_{decomp}$ and RoBERTa* IR-ORA-D is that BM25$_{decomp}$ ignores operation steps. Thus there's more room in the top k passages for retrieval steps. Further, RoBERTa* IR-ORA-D retrieves 10 passages per subquestion, but in $BM25_{decomp}$ we best try to match the number of passages retrieved by MDR and MEX as further explained in section 4.2.
- **MDR**: Multihop dense retrieval
- **MEX**: New architecture as described in Section 3.3.

However, one thing to note is that recall is defined differently for StrategyQA's baseline and MDR [3] [12]. Thus we've defined three slightly different recall metrics as below. Note that "My recall" is the least forgiving while "MDR recall" is the most.

For the formal mathematical definitions, refer to the following notation

- Q: the set of all questions in the dataset
- R_q: the set of retrieved passages for question q (for the model in question)
- $G_{q,i}$: the set of gold standard passages for question q annotated by annotator i. Note for StrategyQA i goes from 1 to 3.
- I_S: Indicator random variable that is 1 if the set S is not empty, and 0 otherwise.

My recall: % of gold passages (by all annotators) covered by retrieved passages, averaged across all questions

$$myrecall = \frac{1}{|Q|} \times \sum_{q \in Q} \frac{|R_q \cap (\cup_{i \in \{1,2,3\}} G_{q,i})|}{|\cup_{i \in \{1,2,3\}} G_{q,i}|} \tag{4.1}$$

"STQA recall": % of gold passages covered by retrieved passages for the best annotator, averaged across all questions. (Like "my recall" except it's the highest out of gold passages from each annotator)

$$STQArecall = \frac{1}{|Q|} \times \sum_{q \in Q} \max_{i \in \{1,2,3\}} (\frac{|R_q \cap G_{q,i}|}{|G_{q,i}|}) \tag{4.2}$$

"MDR recall": % of questions that have at least one true passage covered

$$MDRrecall = \frac{I_{R_q \cap (\cup_{i \in \{1,2,3\}} G_{q,i})}}{|Q|} \tag{4.3}$$

4.2 Hyperparameters

There are two main hyperparameters of note: *beam size* and *topk*.

topk refers to the top number of paths or passages outputted by the model. This obviously affects recall metrics since the larger *topk* is, the better the recall. The reason we say "paths *or* passages" is because MDR and MEX return paths, but BM25 and $BM25_{decomp}$ return passages.

$BM25$ and $BM25_{decomp}$ score each potential passage p for a question (or subquestion for $BM25_{decomp}$) q as

$$score(p) = BM25(q, p) \tag{4.4}$$

Then we return the top k of the passages out of all the passages in the corpus.

4.2.1 Paths

For MDR, each path is composed of 2 passages since the authors assume a question decomposition of length 2. For MEX, the path for query q is composed of d_q passages where d is the length of the question decomposition for query q. Note that both MDR and MEX score the *paths* as a sum of the scores of the passages in it.

Specifically, consider a path P, an ordered list of passages $\{p_1, p_2, ..., p_{d_q}\}$, and the corresponding query q with a question decomposition made of the subquestions $\{q_1, q_2, ..., q_{d_q}\}$. That means that at hop i, passage p_i was retrieved and the query Q_i was q_i concatenated with the first $i - 1$ passages of P. The path-score that model M gives path P is

$$pathscore(P) = \sum_{i \in 1,...,d_q} M(Q_i, p_i) \tag{4.5}$$

	$Q_{2,1}$	$Q_{2,2}$	$Q_{2,3}$
Q_1	$(p_{1_1}, p_{2_{1,1}})$	$(p_{1_1}, p_{2_{1,2}})$	$(p_{1_1}, p_{2_{1,3}})$
Q_1	$(p_{1_2}, p_{2_{2,1}})$	$(p_{1_2}, p_{2_{2,2}})$	$(p_{1_2}, p_{2_{2,3}})$
Q_1	$(p_{1_3}, p_{2_{3,1}})$	$(p_{1_3}, p_{2_{3,2}})$	$(p_{1_3}, p_{2_{3,3}})$

Figure 4.1: A pictorial representation of the explanation of beam size. This matrix A represents $S_1 \times S_2$ and $A_{i,j}$ is the path generated by the query pair represented by row i and column j of A.

where $M(Q, p)$ is the passage-score (proportional to the probability given in Figure 2.1) that model M gives the passage p given query Q.

The obvious question now, is where do we get the paths from? If we are to return the top k paths, what is the set of paths that we choose from? This is where *beam size* comes into play.

Consider a question q with 2 subquestions. Let *beamsize*$= 3$. Then for hop 1, we construct query Q_1 (which is simply q_1) and get the 3 best passages by passage-score. We then construct a set 3 queries $Q_{2,1} = (q_2, p_{1_1})$, $Q_{2,2} = (q_2, p_{1_2})$, $Q_{2,3} = (q_2, p_{1_3})$. For each query $Q_{2,i}$, we get the 3 best passages by passage-score $\{p_{2_{i,1}}, p_{2_{i_2}}, p_{2_{i,3}}\}$. Then we choose the best path $(p_{1,i}, p_{2_{i,j}})$.

For the sake of explanation, define a set S_i that consists of all the queries at hop i. In this case

$$S_1 = \{Q_1\} \tag{4.6}$$

$$S_2 = \{Q_{2,1}, Q_{2,2}, Q_{2,3}\} \tag{4.7}$$

Now consider every pair of queries in $S_1 \times S_2$. For each pair, each query brings in one passage, thus each pair gives us a path of length 2. Now we have $|S_1| \times |S_2|$ paths to choose from and we return the top k as defined by the score function in equation 4.5.

This is pictorially shown in Figure 4.1.

To generalize this to more than 2 hops, we do the same calculations, but consider every element in $S_1 \times S_2 \times ...S_{d_q}$.

One thing to note is that if we use the same *topk* to compare passage-scoring models (BM25 and Bm25$_{decomp}$) and path-scoring models (MDR and MEX), it's an unfair comparison since the number of unique passages retrieved may be different. For some *topk*=k, the top k paths for a query q could include anywhere between k and $k \times d_q$ unique passages between them. Thus, as seen in Table 4.1, we found the number of unique passages k' retrieved across k top paths (for all relevant k), averaged across all questions. Then for passage-scoring models, we used the respective k' for the *topk* hyperparameter.

One takeaway from the results in Table 4.1 is that the number of unique passages retrieved for the top k paths is much closer to k than $k \times d_q$. Note that we only have the *average* number of unique passages retrieved across all questions, but we know that all questions in the StrategyQA dataset have at least 2 hops, so we can be confident that for most $q \in Q$, the above statement holds.

topk	model	**num. unique passages retrieved**
5	MDR	**6**
5	MEX	**7**
10	MDR	**10**
10	MEX	**11**

Table 4.1: Number of unique passages retrieved, rounded to the nearest whole number, for different configurations of model and *topk*

Model	My recall	STQA recall	MDR recall
BM25	6.9%	13.0%	21.0%
BM25$_{decomp}$	15.4%	27.1%	40.4%
MDR	14.0%	24.3%	38.0%
MEX	11.8%	21.1%	34.1%

Table 4.2: Results for all 4 models described in Section 4.1 for the configuration of *beamsize, topk*=5 passages/paths

Thus that means there's a lot of overlap of passages between the k paths.

Hypothetically, increasing *beam size* could improve our results since it would increase the number of paths to choose from. There is a possibility that the best overall path $p*$ out of all possible paths would not be found with a *beam size* too low (if the first few passages of $p*$ score low but lead to high scoring passages later in the path). However, the overlap of passages shows that this is unlikely and increasing *beam size* is not an effective or efficient way significantly improving results.

4.3 Experiments

For each of the 4 models described in section 4.1, we consider two configurations of *topk* (paths/passages): 5 and 10.

For *top5*, see 4.2.

For *top5*, see 4.3.

You may notice that the STQA recall for BM25 and BM25$_{decomp}$ is slightly different than the results reported in Table 3.1. This may be due to small configuration or rounding differences.

We also wrote an oracle model for MEX. Instead of using the previously *retrieved* passages as the context at each hop, we followed each annotator's path and used the *gold* passage they specified (if it was not an operation step).

Note that aren't any paths constructed for MEX-oracle, since we follow the single path of each annotator. Thus for each subquestion at annotator path, we retrieve the top k passages for some set k. We tuned k such that the number of uniquely retrieved passages is closest to 11,

Model	My recall	STQA recall	MDR recall
BM25	10.4%	18.1%	29.7%
BM25$_{decomp}$	20.2%	34.0%	52.1%
MDR	19.4%	32.4%	49.3%
MEX	15.3%	26.6%	42.8%

Table 4.3: Results for all 4 models described in Section 4.1 for the configuration of *beamsize, topk*=10 passages/paths

type	**recall**
my recall	14.3%
STQA recall	25.2%
MDR recall	40.2%

Table 4.4: Results for MEX-oracle where $k = 11$.

as seen in Table 4.4. We chose a k which resulted in about 11.6 uniquely retrieved passages forMEX-oracle.

Note that the results for MEX-oracle are comparable to that in Table 4.3, despite the fact that at each hop, the oracle uses gold-standard contexts. This is interesting because it shows that the retrieval capabilities in MEX are up to par with the oracle at each hop.

4.3.1 Error Analysis

In order to better understand the differences between the results from MDR on StrategyQA versus MEX, we performed some error analysis. An overall view is seen in Table 4.5.

We also performed the same analysis for MEX versus our best model, BM-25$_{decomp}$, which is seen in Table 4.6.

The first thing to note is that BM25$_{decomp}$, MDR, and MEX all have their own strengths, since none totally eclipses the other. However, note that when we switch to the *top10* configuration from the *top5* configuration, the number of gold passages found by MDR and BM25$_{decomp}$ increases significantly, and for BM25$_{decomp}$, it nearly doubles. However, there is not as much of a significant rise for MEX. While a possible explanation is that as k increases, the overlap of

topk	**type**	**num gold passages found**
top5	found by both	30
	found only by mdr	83
	found only be MEX	67
top10	found by both	57
	found only by mdr	102
	found only be MEX	69

Table 4.5: Error analysis statistics for MDR vs. MEX

topk	type	num gold passages found
top5	found by both	32
	found only by BM25$_{decomp}$	94
	found only be MEX	64
top10	found by both	51
	found only by BM25$_{decomp}$	117
	found only be MEX	74

Table 4.6: Error analysis statistics for MEX vs. BM25$_{decomp}$

unique passages between paths increases more for MEX compared to MDR or BM25$_{decomp}$, Table 4.1 debunks this idea. MEX increases in the number of unique passages retrieved by around the same factor as MDR. Another explanation could be that some of the spots for the top passages retrieved by MEX are being taken up by irrelavent passages. This is discussed further in section 4.3.1.

Fine-grain subset analysis

In this section, we outline a few samples for more fine-grain error analysis. For example consider the question ”Will the Albany in Georgia reach a hundred thousand occupants before the one in New York?” with the following subquestions

- ”What is the population of Albany, Georgia?”,
- ”What is the population of Albany, New York?”,
- ”What is the difference between 100,000 and #1?”,
- ”What is the difference between 100,000 and #2?”,
- ”Is #3 smaller than #4?”

The dense methods MDR, MEX, and MEX-oracle all retrieve passages about Albany, Georgia and Albany, New York, but also retrieve passages about Albany, Illinois and Albany, Oregon. Meanwhile, the sparse methods only retrieve passages about Albany, Georgia and Albany, New York. This corroborates the idea that spare methods do better when there is higher lexical overlap between query and gold standard passages.

Something else we noticed is that for MEX and MEX-oracle, some of the top spots are being taken up by passages like ”Clandestine Human Intelligence” and ”Military intelligence-31”. These passages get retrieved multiple times across multiple questions, leading us to believe that operation/operation-adjacent hops like hops 3, 4, and 5 in the decomposition above with numerical values cause the wrong passages to be retrieved, taking up spots in the top k. The solution here in the future would be to be harsher about what hops count as pure retrieval versus operation.

On the other hand, there are cases where MEX captures passages that other models don’t. For example, consider the question ”Is greed the most prevalent of the Seven Deadly Sins?” with the following decomposition

- ”Is greed a deadly sin?”
- ”Is gluttony a deadly sin?”
- ”What percent of crimes involved greed?”
- ”What percent of people are overweight?”
- ”If #1 and #2 are both yes is #3 a higher percentage than #4?”

MEX captures the gold passage ”Gluttony-2” that MDR doesn’t capture. It can easily be seen why since ”gluttony” is in the question decomposition. Further, while the gold passages contain passages for the other seven deadly sins, MEX only captures the one for gluttony since the other sins aren’t explicitly in the decomposition. This is a prime example of how the performance of MEX is completely dependent on the quality of the decomposition.

4.3.2 Discussion

Across all types of recall and across both configurations, the order in performance from best to worst is BM25$_{decomp}$, MDR, MEX, then BM25.

This differs from our prediction. Firstly, we expected the dense models, MDR and MEX, to perform better than the sparse models, BM25 and BM25$_{decomp}$. This would have been in accordance with [5]. Secondly, we expected MEX to perform better than MDR because MEX had access to the explicit question decomposition as explained in Section 3.3.

From their findings, [12] hypothesized that explicit question decompositions aren’t as helpful for dense methods. Our results from MEX and MEX-oracle corroborate this idea and the results from [11], which, seemingly contrarily, experienced improvements in using explicit question decompositions, but did so while using the *sparse* model TF-IDF (similar to BM25). Thus, our work, MDR, and [11] all corroborate the idea the explicit question decompositions are useful for sparse retrieval models like BM25 and TF-IDF, but not for dense models like MDR.

To explain why the sparse models did overall better than dense models, we note that according to the paper that introduced BM25, their results show that dense approaches don’t *always* outperform sparse approaches. Especially for questions that require high lexical overlap detection, BM25 outperforms dense models like BERT-based models [8]. To test if this may be the case, for our setup, we check the lexical overlap between the subquestions of StraqtgyQA and gold standard passages against the overlap between the main questions of StrategyQA and the gold standard passages. Our overlap metric is calculated by counting the number of tokens from the query question or set subquestions that overlap with any gold standard passage. This is averaged across the all questions. Our results can be seen in 4.7. While the difference is small, the subquestions overall have a higher lexical overlap with the gold standard passages which could contribute to why BM25$_{decomp}$ performed better than dense methods on StrategyQA.

	overlap
subquestions	60.4%
questions	57.4%

Table 4.7: Lexical overlap of subquestions with gold standard versus that of questions with gold standard passages for StrategyQA.

4.4 Further Questions and Improvements

4.5 Limitations

4.5.1 Question Decompositions

It's important to note that the question decompositions we use are human annotated. Thus our results are limited by the quality of these decompositions as seen in 4.3.1. It seems unlikely if the decompositions provided are the only way to decompose that given question. Further, the question still remains of how good do these question decompositions need to be to replicate our results?

4.5.2 Evidence Matching

Similarly, it's unclear if the gold standard passages are the only and/or best passages for a given question or sub-question. In order to avoid biased questions that seem targeted or have high lexical overlap with passages, question writers write the question and decompositions without seeing the corpus. As described in 3.1, gold standard evidence matching happens after. Thus it remains to be proved that the question can be properly answered by these passages.

4.6 Future Directions

There are many directions to take this work in. Our main suggestion, corroborated by the results in [5], is a joint sparse and dense model. As seen in in 4.3.1, BM25$_{decomp}$, a sparse model, and MDR/MEX retrieve distinct passages. A weighting scheme between BM25 and dense retrieval at every hop would greatly benefit our results. Further, instead of simply retrieving BM25 at each hop as in BM25$_{decomp}$, a better solution would be to mimic the context-concatenation of MDR/MEX . In other words, the query at hop i should be the concatenation of subquestion i and the previously retrieved passages, whether they are by BM25 or dense inner product scored.

Another way to improve retrieval would be to finetune the MEX on the StrategyQA dataset. We compiled negative passages for intelligent training, as in [4], with incorrect passages retrieved by BM25 and MDR, but were limited by resources for finetuning. Finetuning with this and in-batch negatives would surely improve retrieval.

[12] were able to see significant improvements when increasing k in *topk* and after re-ranking the retrieved passages. Both would be an efficient way to improve MEX. In fact, retrieval for

MDR shot up after $k = 20$, but we were too limited by resources to try this.

It would also be interesting to evaluate MEX on other datasets such as BREAK from [11], a new annotation of HotpotQA with decompositions. This would allow for a good comparison with MDR which was evaluated in HotpotQA.

Finally, as discussed in Section 4.3.1, being harsher about what hops count as retrieval would help improve performance as well.

Bibliography

[1] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *CoRR*, abs/1704.00051, 2017. URL `http://arxiv.org/abs/1704.00051`. 1

[2] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 1156–1165, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450329569. doi: 10.1145/2623330.2623677. URL `https://doi.org/10.1145/2623330.2623677`. 2.3

[3] Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? A question answering benchmark with implicit reasoning strategies. *CoRR*, abs/2101.02235, 2021. URL `https://arxiv.org/abs/2101.02235`. (document), 1, 2.3, 3.1, 3.2, 3.3, 4.1

[4] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. *CoRR*, abs/2004.04906, 2020. URL `https://arxiv.org/abs/2004.04906`. 2.1, 2.2, 3.2, 4.6

[5] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. *CoRR*, abs/1906.00300, 2019. URL `http://arxiv.org/abs/1906.00300`. 2.1, 3.2, 4.3.2, 4.6

[6] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL `http://arxiv.org/abs/1907.11692`. 2.2, 3.2

[7] Fabio Petroni, Tim Rocktäschel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. Language models as knowledge bases? *CoRR*, abs/1909.01066, 2019. URL `http://arxiv.org/abs/1909.01066`. 2.3

[8] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3:333–389, 01 2009. doi: 10.1561/1500000019. 3.2, 4.3.2

[9] Alon Talmor and Jonathan Berant. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of*

the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 641–651, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1059. URL `https://aclanthology.org/N18-1059`. 1

[10] Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. Constructing Datasets for Multi-hop Reading Comprehension Across Documents. *arXiv e-prints*, art. arXiv:1710.06481, October 2017. 1

[11] Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. Break it down: A question understanding benchmark. *Transactions of the Association for Computational Linguistics*, 8:183–198, 2020. doi: 10.1162/tacl_a_00309. URL `https://aclanthology.org/2020.tacl-1.13`. 2.3, 4.3.2, 4.6

[12] Wenhan Xiong, Xiang Lorraine Li, Srinivasan Iyer, Jingfei Du, Patrick S. H. Lewis, William Yang Wang, Yashar Mehdad, Wen-tau Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oguz. Answering complex open-domain questions with multi-hop dense retrieval. *CoRR*, abs/2009.12756, 2020. URL `https://arxiv.org/abs/2009.12756`. (document), 2.2, 2.3, 2.3, 2.2, 4.1, 4.3.2, 4.6

[13] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *CoRR*, abs/1809.09600, 2018. URL `http://arxiv.org/abs/1809.09600`. 2.2

[14] Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. Retrieving and reading: A comprehensive survey on open-domain question answering. *CoRR*, abs/2101.00774, 2021. URL `https://arxiv.org/abs/2101.00774`. 1