Interpretable Multi-Step Reasoning with Knowledge Extraction on Complex Healthcare Question Answering

Ye Liu[1], Shaika Chowdhury[1], Chenwei Zhang[2], Cornelia Caragea[1], Philip S. Yu[1]
[1]Department of Computer Science, University of Illinois at Chicago, IL, USA
[2]Amazon, Seattle, WA, USA
yliu279@uic.edu, schowd21@uic.edu, cwzhang@amazon.com, cornelia@uic.edu, psyu@uic.edu

ABSTRACT

Healthcare question answering assistance aims to provide customer healthcare information, which widely appears in both Web and mobile Internet. The questions usually require the assistance to have proficient healthcare background knowledge as well as the reasoning ability on the knowledge. Recently a challenge involving complex healthcare reasoning, HeadQA dataset, has been proposed, which contains multiple choice questions authorized for the public healthcare specialization exam. Unlike most other QA tasks that focus on linguistic understanding, HeadQA requires deeper reasoning involving not only knowledge extraction, but also complex reasoning with healthcare knowledge. These questions are the most challenging for current QA systems, and the current performance of the state-of-the-art method is slightly better than a random guess. In order to solve this challenging task, we present a **Mu**lti-step **r**easoning with **K**nowledge **e**xtraction framework (**MurKe**). The proposed framework first extracts the healthcare knowledge as supporting documents from the large corpus. In order to find the reasoning chain and choose the correct answer, **MurKe** iterates between selecting the supporting documents, reformulating the query representation using the supporting documents and getting entailment score for each choice using the entailment model. The reformulation module leverages selected documents for missing evidence, which maintains interpretability. Moreover, we are striving to make full use of off-the-shelf pretrained models. With less trainable weight, the pretrained model can easily adapt to healthcare tasks with limited training samples. From the experimental results and ablation study, our system is able to outperform several strong baselines on the HeadQA dataset.

KEYWORDS

Complex Healthcare Reasoning, Knowledge Retrieval, Multi-Step Reasoning, Query Reformulation

ACM Reference Format:
Ye Liu[1], Shaika Chowdhury[1], Chenwei Zhang[2], Cornelia Caragea[1], Philip S. Yu[1]. 2020. Interpretable Multi-Step Reasoning with Knowledge Extraction on Complex Healthcare Question Answering. In *Conference '20, October, 2020.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456

Conference '20, October, 2020, New York, USA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Neural network models have achieved great success on the recent progress of question answering (QA). In some of the popular datasets, such as SQuAD [32] and bAbI [43], the machine can achieve near human-level performance. However, these datsets are easy for machine since the context contains the answer and often surface-level knowledge is sufficient to answer [44]. The recently released HeadQA [39] is an ambitious test for AI systems. This dataset consists of 6,765 multiple choice questions authored for college students in the healthcare area to have the specialization license. The dataset contains question (containing context) with four or five candidate option choices from 6 categories including Medicine, Pharmacology, Psychology, Nursing, Biology and Chemistry. A small percentage ($\sim$ 14%) of the Medicine questions refer to images, that provide additional information to answer correctly. These questions require sophisticated reasoning and language understanding abilities to be answered correctly, and even for humans (i.e., medical college students), these questions take over a period of one year or more for them to pass the exam.

Compared to basic reading comprehension based QA setup where the answers to a question are usually found in the given small context, the HeadQA setup needs to extract relevant knowledge according to the context and question. Another characteristic of this dataset is that unlike current datasets like TRIVIAQA-open [18], SQUAD-open [32] and ARC [8], the gold document and relevant search document set are not provided for each question. This makes HeadQA represent a unique obstacle in the QA as the system now needs to search for relevant documents from the whole Wikipedia corpus. The performance of current state-of-the-art methods is only slightly better than random guess [39]. The performance degradation is mainly from failing to retrieve the relevant documents for the question answering model [16].

In previous works, single-step retrieve-and-read question answering (QA) systems [5] failed to perform well on complex questions dataset [8, 47] as the question does not contain sufficient retrievable clues and, thus, all the relevant context cannot be obtained in a single retrieval step [31]. The recently popular multi-hop based QA systems, like the HotpotQA [47] and WikiHop [42], are designed such that they require to reason with information taken from more than one document in order to arrive at the correct answer. The reasoning chains in HotpotQA are well-designed by human; specifically, these datasets assume supporting documents are already obtained and the reasoning chain is generated by human along with the dataset. However, the HeadQA is a more natural

dataset as it is collected from the healthcare specialization exam, so the reasoning chains are unknown and the model needs to extract the relevant healthcare knowledge by itself.

The above mentioned differences lead to multiple challenges for HeadQA. First, finding the relevant supporting documents from the large corpus, like Wikipedia is a challenge, especially since standard IR approaches can be misled by distractions. Second, finding the multi-hop reasoning chain among [47] the plentiful documents is another challenge, since their reasoning path is unclear and thus would need to have a well understanding between the natural language texts. To solve these challenges, our proposed model, **Mu**lti-step **r**easoning with **K**nowledge **e**xtraction framework (**MurKe**), solves this problem in two steps: extract relevant knowledge and reason with the background knowledge. The relevant knowledge extraction aims to narrow the document search space from the whole Wikipedia to the relevant document set by using a combination of token-level and semantic-level retrieval. In the reasoning step, it is possible that the answer might not be present in the initially selected documents or that the model would need to combine information across multiple documents [25]. Thus, we extend the single-step retrieval to multi-step iterative selection, reformulation and entailment module. Given an input question, the selection module finds the most relevant document with the current question. The selected relevant document is sent to the reformulation module which aims to find the guided clue from the selected document and reformulate the current question. For this purpose, we use a reformulation module that is equipped with extractive reading-based attention to reformulate the question. The important pieces of the selected document are highlighted by what we call a reading-answer attention and integrated into a representation of the question via our reformulation module. This new question vector is then used by the selection model to re-rank the context. In this way, it allows the model to select new documents and combine evidence across multiple documents, which could provide the interpretability of the reasoning path. Moreover, the input of the reformulation and entailment module is the same and they can be processed in parallel. Therefore, our method is still efficient, even if the method needs to do iterative several steps.

The main contributions of this paper are: (a) a combination of token-level retrieval and semantic-level retrieval to settle down the search space as a small but sufficient document set, (b) an efficient and effective iterative retrieval-reformulation-entailment framework capable of complex healthcare reasoning, (c) a natural language question reformulation approach that guarantees interpretability in the multi-step evidence gathering process, and d) we illustrate the advantages of our model on the HeadQA dataset.

2 RELATED WORKS

2.1 Question Answering with Knowledge Extraction

Performing question-answering need knowledge extraction setting is far more challenging than its counterpart closed-domain setting as in the latter case the answer can be extracted from a preselected passage [41]. For example, although the recently released AI2 Reasoning Challenge (ARC) dataset [8] contains science-related questions, which also require powerful knowledge and reasoning, it has accompanying ARC corpus with relevant science sentences. As a result, in comparison to HeadQA, it is easier to find answers to ARC questions as the former requires large-scale search to find supporting documents, alongside a reading comprehension module to generate the answer. QA with knowledge extraction originally found answers in the large corpus of unstructured texts [5], but over the years many works have explored QA from knowledge bases (KB) such as Freebase [3] or DBPedia [1]. However, the main drawbacks of KBs are that they are incomplete as well as expensive to construct and maintain [16]. This has rendered free-corpus such as Wikipedia as the preferred choice for knowledge source to provide additional evidence in answering questions, not to mention that it also provides up-to-date information [5]. Most pipelines base their information retrieval module returning the relevant documents on standard IR mechanisms (e.g., TF-IDF), which can fail to contain the correct answer in the ranked documents [24].

2.2 Multi-Step Datasets and Reasoning

Instead of getting the answer from a single context, the questions in the multi-step datasets need to locate multiple contexts to get the answer. [18] developed TriviaQA containing question-answer pairs with several associated evidence documents, which requires inference over multiple sentences to answer correctly. Answering questions in the bAbI dataset [43] requires combining multiple disjoint evidence in the context, however, as the text is synthetic, it fails to completely resemble the complexity of passage structures in human-generated texts [2]. The WikiHop dataset [42] requires more than one Wikipedia document to answer. More recently, the HotpotQA dataset [47] has gained traction in this direction. It contains crowdsourced questions with more diverse syntactic and semantic features [17]. A sequential approach is followed by Memory Network-based models to iteratively store the information gathered from passages in a memory cell [22, 34, 38]. Works by [4, 11, 36] use graph convolutional network [21] to do multi-hop reasoning. Reasoning chains fed into a BERT-based QA model is proposed by [6]. Although much progress has been made with large-scale reasoning datasets [40, 46], these datasets contain gold document contexts corresponding to each question. When it comes to performing multi-step reasoning they still lag behind in performance [47]. Compare to those datasets, HeadQA is more difficult due to it does not have any relevant gold documents for each question and the reasoning chain is unknown.

2.3 Query Reformulation

One direction of query reformulation works on reformulating queries by rewriting the query or only retaining their most salient terms. By selecting important terms from the retrieved document, Nogueira et al. [28] uses reinforcement learning to reformulate the query to maximize the number of relevant documents retrieved. Beyond just selecting important terms, work by [26] refine the query in a well-formed way. In the multi-choices question answering domain, [27] use a sequence to sequence model to retain the most salient term and use an entailment model to set scores to each choice.

Instead of reformulating the explicit query, another direction works on reformulating the latent query vector. Work by [30] showed that query refinement is effective for IR in the bio-medical

domain. [10, 13] turns the query reformulation to the multi-step setting, that retrieval and reader model iterative work. The reader model sets the document latent representation and uses that to reformulate query latent representation. Our work is in the latent vector reformulation direction which is more flexible and also contains the interpretation.

3 TASK DEFINITION

In complex healthcare reasoning, we are given a question $\mathbf{Q}$ containing m tokens $\mathbf{Q} = [q_1, q_2, ..., q_M]$ with a context description $\mathbf{C} = [c_1, c_2, ..., c_L]$ in the question where $L < M$. In some cases, an image $\mathbf{I}$ is provided which contains information related to the question $\mathbf{Q}$. The option set has h candidate option choices $O = \{\mathbf{O}_1, \mathbf{O}_2, ..., \mathbf{O}_h\}$, where each candidate option is a text with R tokens $\mathbf{O}_i = [o_1, o_2, .., o_R]$. The goal is to select the correct answer $\mathbf{A}$ from the candidate option set. For simplicity, we denote $\mathcal{X} = \{\mathbf{Q}, O, \mathbf{I}\}$ as one data sample and denote $y = [y_1, y_2, \ldots, y_h]$ as a one-hot label, where each $y_i = 1(\mathbf{O}_i = \mathbf{A})$ is an indicator function. In the training, N sets of $(\mathcal{X}, y)^N$ are given and the goal is to learn a model $f : \mathcal{X} \rightarrow y$. In the testing, we need to predict y^{test} given test samples $\mathcal{X}^{test}$.

We observe that the context itself is unable to provide enough clues to the correct answer. Hence, we seek to bring supporting knowledge for each data sample from the open knowledge base, like Wikipedia. In this work, our proposed **MurKe** model first extracts the supporting document as the background knowledge, and then finds the reasoning path among them. **MurKe** extracts the question-related supporting documents set $\mathcal{D}_N = \{\mathbf{D}_1, ..., \mathbf{D}_K\}$ from the Wikipedia corpus $\mathcal{D}$ [1]. Note that the extracted relevent document set $\mathcal{D}_N$ comes from a large corpus of documents $\mathcal{D}$, where $|\mathcal{D}| \gg |\mathcal{D}_N|$. Then the document set $\mathcal{D}_N$ is used as external background knowledge to predict the answer option $\mathbf{O}_i \in O$. By using the supporting document from the first part, we design the iterative selection, reformulation and entailment models to find the latent multi-step reasoning chain.

4 THE PROPOSED FRAMEWORK (MURKE)

Since there are no correct search documents for each question in the HeadQA dataset, to get the background knowledge the model needs to search the supporting documents from the whole Wikipedia. However, this is computationally expensive and time-costing. To solve this problem, we need to settle down the space of the supporting documents such that they can cover the information of the question as much as possible, while keeping the size of the supporting document set acceptable enough for downstream processing.

After getting the supporting documents, **MurKe** mimics how humans answer the complex question using background knowledge. Namely, based on the question and extracted supporting knowledge, humans first search one relevant background knowledge and decide whether the current background knowledge could answer the question. If it could, they get the answer. Otherwise humans change the current question according to the first and search a new background knowledge. Similarly, **MurKe** seeks to find the latent reasoning path by selecting top-1 relevant document for the current question, reformulating the current question using the relevant document and, at the same time, performing textual entailment between the top-1 relevant document and the current question with the answer.

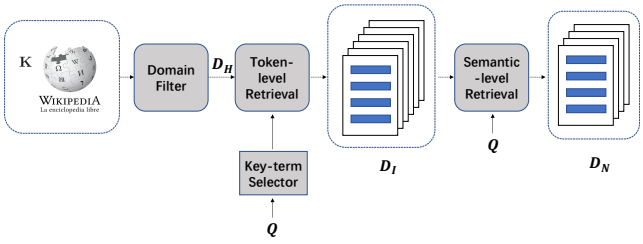

Figure 1: The pipeline of the knowledge extraction step of MurKe model.

4.1 Knowledge Extraction

To provide supporting documents for each healthcare question, we first introduce an effective and efficient preprocessing method to narrow the supporting document space from the whole Wikipedia to the relevant document, as shown in Fig. 1. Since the document scale is few hundred million, only using the semantic-level retrieval is both computation-costing and time-costing. Therefore, we use the token-level retrieval first to settle down the document number, and then use the semantic-level retrieval to further select the relevant documents. A question-based document ranking approach is employed to retrieve the most relevant supporting documents to a given question from a knowledge source Wikipedia $\mathcal{D}$. Since all the questions in our dataset are healthcare science-related, we filter the documents as the categories of depth 4 under "Health" topic [2]. We refer to this corpus of extracted Wikipedia documents as the "WikiHealth" corpus $\mathcal{D}_H$.

Token-level Retrieval To begin with, we use a combination of the neural keyword matching method and TF-IDF method to narrow down the search document scope from "WikiHealth" corpus $\mathcal{D}_H$ down to a set of question related documents $\mathcal{D}_I$. The focus of this step aims to efficiently select a candidate set that can cover the information as much as possible while keeping the size of the set acceptable enough for downstream processing.

Specifically, following Musa et al. [27], we treat the key-term selector as an encoder-decoder model, which inputs the question $\mathbf{Q}$ and answer choices $\{\mathbf{O}_1, \mathbf{O}_2, ..., \mathbf{O}_h\}$, and then outputs the key-terms $\mathbf{T}_Q \in \mathbf{Q}$ and $\{\mathbf{T}_{O_1}, \mathbf{T}_{O_2}, ..., \mathbf{T}_{O_h}\}$ for the question and option choices respectively. Each question forms h new queries by appending each candidate answer option to the question, $\{[\mathbf{T}_Q, \mathbf{T}_{O_1}], [\mathbf{T}_Q, \mathbf{T}_{O_2}], ..., [\mathbf{T}_Q, \mathbf{T}_{O_h}]\}$. The BM25 scoring mechanism [3] is then used to retrieve the *top*-100 (tested in the experiment) possibly relevant documents to each question from the WikiHealth corpus $\mathcal{D}_H$. The search document space of question $\mathbf{Q}$ is the unit document of the retrieved documents from the h queries, represented as $\mathcal{D}_I$.

Semantic-level Retrieval After obtaining the related documents $\mathcal{D}_I$ using token-level retrieval, we seek to use the semantic-level retrieval to further narrow down the documents. The outputs of the neural model are treated as the relatedness score between

[1] https://dumps.wikimedia.org/

[2] https://github.com/attardi/wikiextractor/blob/master/categories.filter

[3] https://radimrehurek.com/gensim/summarization/bm25.html

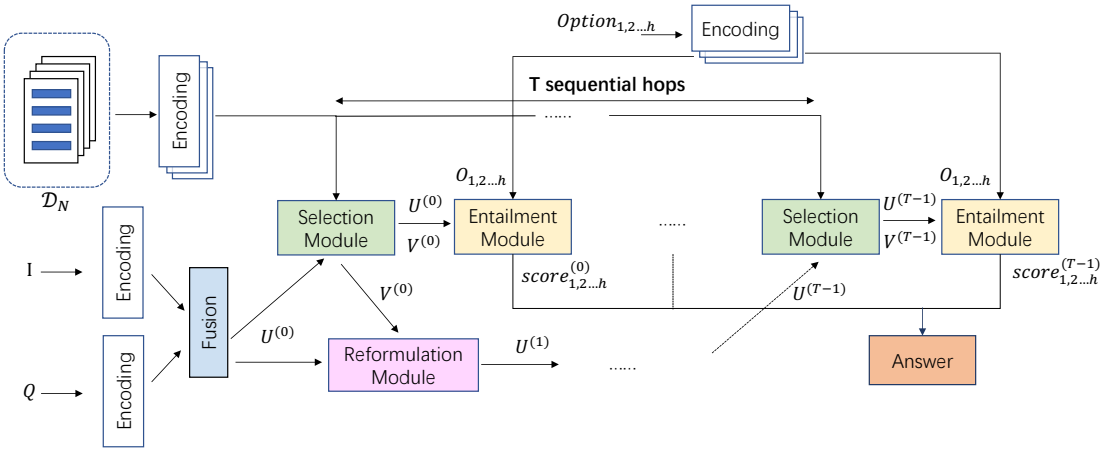

Figure 2: The framework of the reasoning step of MurKe model. Selection module executes a question to obtain the most relevant documents (Sec. 4.2.1). The selected document is sent to both reformulation and entailment module at the same time. Reformulation module refines a question by considering the important information of selected documents (Sec. 4.2.2). Entailment module uses selected documents as evidence to compute probabilities for each choice (Sec. 4.2.3) The final decision is made over time to determine the final answer (Sec. 4.2.4).

the input question and the documents. The scores will be used to sort and limit all the upstream documents. This step, as shown in Fig. 1, aims to screen $\mathcal{D}_I$ to the semantic-level related documents $\mathcal{D}_N$, which can be helpful for the downstream modeling.

From these retrieved documents, to find the document that is the most semantically related to each question, we use the language model BERT [12] which is pretrained on biomedical corpora, namely BioBERT [23]. We use the special token to concatenate the question and document together as the input to the BERT model:

$$[CLS]\ Question\ [SEP]\ Document\ [SEP] \tag{1}$$

We applied an affine layer and sigmoid activation on the last layer output of the $[CLS]$ to get the scalar value. Subsequently, the documents of each question whose value is above the document relevance threshold th_r is considered as the search documents input to our model. In our experiment, we set the threshold th_r as 0.9. In the end, we can get the semantic-level related documents $\mathcal{D}_N$ for each of the search question $\mathbf{Q}$.

4.2 Iterative Multi-Step Reasoning

After getting the supporting documents $\mathcal{D}_N$, in this section, **MurKe** proposes three modules - selection module, reformulation module and entailment module - which work iteratively to find the latent multi-step reasoning path. The reasoning diagram of the **MurKe** framework is shown in Fig. 2. Specifically, **MurKe** contains three sub-networks: Selection module first computes a relevance score for each document with regard to a given question and ranks them according to the score. The top one document is sent to both reformulation and entailment module. Reformulation module uses reading-answer attention to extract relevant information from top one selected document in order to update the latent representation of the question. At the same time, entailment module computes the entailment score of each candidate option and the selected document to get the final answer. Since the update of the reformulation conditions on the result of the selection module and the reformulated question can help get the confidence of the candidate choices in the following entailment model, this provides a way for the multi-step interaction between search engine (selection) and the matching model (entailment) to communicate with each other. Moreover, the model is processing very fast as the entailment model and reformulation model can be processed in parallel.

4.2.1 Selection Module. The selection module computes a relevance score between each related document and the given search question, which is represented in Fig. 3 a). The related document representations are computed independently of the question and once computed, they are not updated. The relevance score of a related document is computed as an inner product between the related document and the question vectors. The related document and question representations are computed as follows.

Given a document $\mathbf{D} = [\mathrm{d}_1, \mathrm{d}_2, ..., \mathrm{d}_\mathrm{N}]$ in the relevant document set $\mathcal{D}_N$ of question P consisting of N tokens, a bidirectional multi-layer GRU (BiGRU) [7] encodes each token in the document $[\mathbf{d}_1, \mathbf{d}_2, ..., \mathbf{d}_N] = \mathrm{BiGRU}([\mathrm{d}_1, \mathrm{d}_2, ..., \mathrm{d}_\mathrm{N}])$, where $\mathbf{d}_j \in \mathbb{R}^{2d}$ is the concatenation of the forward and backward GRU last layer hidden units. The question $\mathbf{Q} = [\mathrm{q}_1, \mathrm{q}_2, ..., \mathrm{q}_\mathrm{M}]$ with M tokens is encoded by another network with the same architecture to obtain the question embedding $[\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_\mathrm{M}] = \mathrm{BiGRU}([\mathrm{q}_1, \mathrm{q}_2, ..., \mathrm{q}_\mathrm{M}])$. To solve the long-term dependencies in the document, we compute the probability distribution α_j depending on the degree of relevance between word and the other words (in its context). The self-attention document vector $\mathbf{E_D} \in \mathbb{R}^{N \times 2d}$ is computed as a weighted combination of all contextual embeddings:

$$\alpha_\mathrm{j} = \frac{\exp\left(\mathbf{w} \cdot \mathbf{d_j}\right)}{\sum_{j'=1}^{N} \exp\left(\mathbf{w} \cdot \mathbf{d_{j'}}\right)}, \quad \mathbf{E_D} = W_s \cdot \alpha_\mathrm{j} \cdot \mathbf{d_j} \tag{2}$$

where $\mathbf{w} \in \mathbb{R}^{2d}$ and $W_s \in \mathbb{R}^{2d \times 2d}$, used in the bilinear term, is a learned weight matrix. In the same way, we calculate the $\mathbf{E_Q} \in$

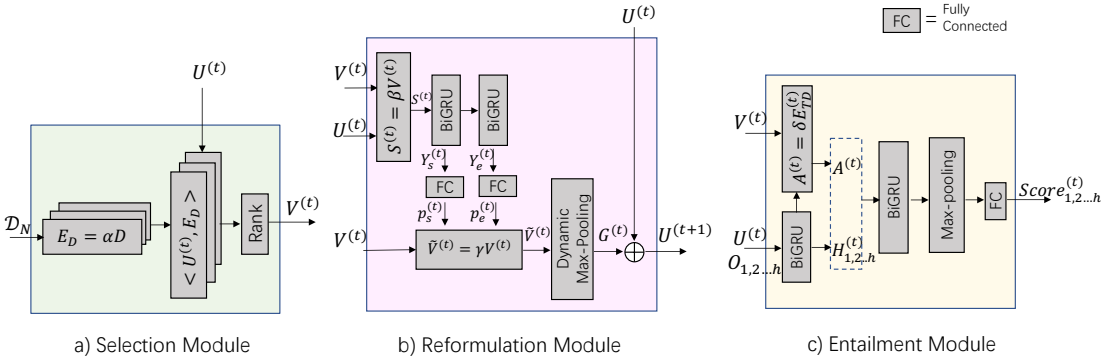

Figure 3: The different building blocks of the proposed end-to-end trainable model.

$\mathbb{R}^{M\times 2d}$ as the question embedding. As some queries may contain image $\mathbf{I}$, so we fuse the hidden states of question embedding and the corresponding image embedding to the initial question vector as $\mathbf{U}^{(0)} = \text{Fuse}(\mathbf{E}_Q, \mathbf{E}_I)$. We tried different fusion methods, such as con(), avg() and use bilinear module bil(), which are discussed in the experiment section.

The relevance score of a document with regard to the question (score($\mathbf{U}^{(t)}, \mathbf{E}_D$)) is computed by a simple inner product $\langle \mathbf{U}^{(t)}, \mathbf{E}_D \rangle$, where t means t-th iteration. The document retriever ranks all the documents in $\mathcal{D}_N$ and sends the embedding of top-1 scoring document $\mathbf{E}^{(t)}_{D_{top1}}$ to the following modules, reformulation and entailment. For notation simplicity, we use $\mathbf{V}^{(t)}$ instead to refer to the top-1 scoring document $\mathbf{E}^{(t)}_{D_{top1}}$.

4.2.2 Latent Question Reformulation Module. The latent question reformulation aims to find some evidence from the selected document so that combining the evidence with the current question representation formulates a new representation that can help answer more correctly, as shown in Fig. 3 b). The reformulated question is sent back to the entailment and retriever, which uses it to calculate the entailment score between hypothesis and premise and re-rank the documents in the corpus, respectively. More formally, a reformulation module takes the encoding of the top one selected document from the previous selection module, $\mathbf{V}^{(t)}$, and the previous representation of the question, $\mathbf{U}^{(t)}$, as input, and produces an updated reformulation of the question $\mathbf{U}^{(t+1)}$. Moreover, in order to provide the interpretability of the model, we want to extract sub-phrase of the document which can bring the guided clue for the current question. Therefore, we formulate it as a reading comprehension task [33] which aims to find the answer span in the document and use the found answer span to update the question.

Reading-answer Attention: The matching of stop words is presumably less important than the matching of content words. In this step, the goal is to compare the question embedding and the contextual document embeddings and select the pieces of information that are relevant to the question. We plan to learn the reading-based attention of a token as the probability that the predicted span has started before this token and will end after. Therefore, we calculate the question-aware document representation as $\mathbf{S}^{(t)} = \beta_j \mathbf{V_j}^{(t)}$ where, $\beta_j = \text{softmax}_j \sum_i \mathbf{U}_i^{(t)} \mathbf{W}_c \mathbf{V_j}^{(t)}$, where $\mathbf{U}_i^{(t)}$ represents the i-th token in $\mathbf{U}^{(t)}$, $\mathbf{V}_j^{(t)}$ represents the j-th token in $\mathbf{V}^{(t)}$ and the $\mathbf{W}_c$ is the training weight. Following [13], we use the idea of reader module to compute the reading-based attention vector. Given the question-aware document representation $\mathbf{S}(t)$, we compute the starting and ending index position probability $\mathbf{p}_s^{(t)}$ and $\mathbf{p}_e^{(t)}$ using two BiGRUs followed by a linear layer and a softmax operator. They are computed from:

$$\mathbf{Y}_s^{(t)} = \text{BiGRU}\left(\mathbf{S}^{(t)}\right) \qquad \mathbf{Y}_e^{(t)} = \text{BiGRU}\left(\mathbf{Y}_s^{(t)}\right) \tag{3}$$

$$\mathbf{p}_s^{(t)} = \text{softmax}\left(\mathbf{w}_s \mathbf{Y}_s^{(t)}\right) \qquad \mathbf{p}_e^{(t)} = \text{softmax}\left(\mathbf{w}_e \mathbf{Y}_e^{(t)}\right) \tag{4}$$

where w_e and w_s are trainable vectors of $\mathbb{R}^{2d}$. The two probability vectors $\mathbf{p}_s^{(t)} \in \mathbb{R}^{\mathbb{N}}$ and $\mathbf{p}_e^{(t)} \in \mathbb{R}^{\mathbb{N}}$ are not used to predict an answer, but to compute a reading-based attention vector $\gamma^{(t)}$ over the document. Intuitively, these probabilities represent at step t how likely each word is to be the beginning and the end of the answer span respectively. We define the reading-based attention of a token as the probability that the predicted span has started before this token and will end after, which can be computed as follows:

$$\gamma_i^{(t)} = \left(\sum_{k=0}^{i} \mathbf{p}_{s_k}^{(t)}\right)\left(\sum_{k=i}^{N} \mathbf{p}_{e_k}^{(t)}\right) \tag{5}$$

Further, we use these attention values to re-weight each token of the document representation. We compute $\widetilde{\mathbf{V}}_i^{(t)} \in \mathbb{R}^{N\times 2d}$ with:

$$\widetilde{\mathbf{V}}_i^{(t)} = \gamma_i^{(t)} \mathbf{V_i}^{(t)} \tag{6}$$

Dynamic Knowledge Extraction Max-Pooling: This layer aims at collecting the relevant evidences of $\widetilde{\mathbf{V}}_i^{(t)}$ with length N to add to the current question representation with length M. We partition the row of the initial sequence into M approximately equal parts. It produces a grid of $M \times 2d$ in which we apply a max-pooling operator in each window. As a result, a matrix of fixed dimension adequately represents the input, preserving the global structure of the document, and focusing on the important elements of each region. This can be seen as an adaptation of the dynamic pooling layer proposed by Socher et al. [35]. Formally, let $\widetilde{\mathbf{V}}_i^{(t)}$ be the input matrix representation, we dynamically compute the kernal size, w, of the max-pooling according to the length of the input sequence and the required output shape: $w = \lceil \frac{N}{M} \rceil$, $\lceil \cdot \rceil$ being the ceiling

Algorithm 1 Multi-step reasoning with knowledge extraction(MurKe)

1: **Input:** Question $\mathbf{Q}$, Candidate options set $O = \{\mathbf{O}_1, \mathbf{O}_2, ...\mathbf{O}_h\}$, Image $\mathbf{I}$, Wikipedia corpus $\mathcal{D}$, multi-steps T.
2: **Knowledge Extraction:** $\mathcal{D}_N \leftarrow \mathcal{M}.Extractor(\mathbf{Q}, \mathcal{D})$ # Use token-level and semantic-level retrieval get the supporting document
3: Initialize the search query $\mathbf{U}^{(0)} \leftarrow$ Fuse($\mathbf{Q}, \mathbf{I}$), t = 0
4: **while** t < T **do**
5: $\mathbf{V}^{(t)} \leftarrow \mathcal{M}.selection(\mathbf{U}^{(t)}, \mathcal{D}_N, 1)$ # Select the top-1 relative documents from the document set.
6: $\mathbf{U}^{(t+1)} \leftarrow \mathcal{M}.reformulation(\mathbf{V}^{(t)}, \mathbf{U}^{(t)})$ # Use a sequence of token-level attention to extract relevant information and update query.
7: $Score_1^{(t)}, Score_2^{(t)}, ..., Score_h^{(t)} \leftarrow \mathcal{M}.entailment(\mathbf{V}^{(t)}, \mathbf{U}^{(t)}, \{\mathbf{O}_1, \mathbf{O}_2, .., \mathbf{O}_h\})$ # Text entailment to measure the similarity between hypothesis (the retrieved document) and premises (the query with the different choices).
8: t = t + 1
9: **end while**
10: $\mathcal{L}(\mathcal{M}) = -\log(\frac{1}{T}\sum_{t=0}^{t=T} Score_i^{(t)})$ # update module based on loss function
11: **return** Predicted answer option $\hat{y}_i$.

function. Then the output representation of this pooling layer is the extracted knowledge from the document represented as $\mathbf{G}^{(t)} \in \mathbb{R}^{M \times 2d}$ where

$$\mathbf{G}_i^{(t)} = \max_{k \in \{iw, \ldots, (i+1)w\}} \left(\widetilde{\mathbf{V}_k}^{(t)}\right) \tag{7}$$

Finally, the updated representation of the question $\mathbf{U}^{(t+1)} \in R^{M \times 2d}$ is the sum of $\mathbf{U}^{(t)}$ and $\mathbf{G}^{(t)}$.

4.2.3 Entailment Module. Given the selected top-1 document, the module needs to select a particular answer from the candidate options choices $O = \{\mathbf{O}_1, \mathbf{O}_2, ..., \mathbf{O}_h\}$. In the approach pioneered by [20], a multiple reading comprehension problem is converted into an entailment problem wherein each top-1 selected document is a premise and the question combined with each candidate answer is used as a hypothesis, and the model's probability that the premise entails this hypothesis becomes the candidate answer's score.

In our method, the embedding of selected top-1 document $\mathbf{V}^{(t)}$ is treated as the embedding vector of the premise $\mathbf{P}^{(t)}$, while the hypothesis is the combination of question and candidate choices. Since the question representation is vectorized as $\mathbf{U}^{(t)}$ but the candidate choices is still in the token-level, the question embedding $\mathbf{U}^{(t)}$ is treated as the initial hidden states and the choice token $O = \{\mathbf{O}_1, \mathbf{O}_2, ...\mathbf{O}_h\}$ are passed through a BiGRU separately in order to capture the dependency between the words, which returns a new hypothesis representation $\mathbf{H}_{1,2...,h}^{(t)} \in \mathbb{R}^{2d \times (M + len(\mathbf{O}_{1,2...,h}))}$. Then, an attention mechanism is used to determine the attention weighted representation of the j-th word in the premise as follows:

$$e_{ij} = \mathbf{P}_i^{(t)} \cdot \mathbf{H}_j^{(t)},\ \delta_{ij} = \frac{\exp(\mathrm{e_{ij}})}{\sum_{r=1}^{K} \exp(\mathrm{e_{rj}})},\ \mathbf{A}_j^{(t)} = \sum_i \delta_{ij} \mathbf{P}_i^{(t)} \tag{8}$$

The matching layer is a BiGRU($\cdot$) with the input $\mathbf{M}_j^{(t)} = [\mathbf{A}_j^{(t)}; \mathbf{H}_j^{(t)}]$ ([;] is the concatenation operator). Finally, the max-pooling result over the hidden states of the matching model is used for softmax classification to get the entailment score for each choice $\{\mathrm{Score}_1^{(t)}, \mathrm{Score}_2^{(t)}, \cdots, \mathrm{Score}_h^{(t)}\}$. The diagram of entailment module is Fig. 3 c). It is worth to notice that the input of the entailment module and question reformulation module are the selected top-1 document $\mathbf{V}^{(t)}$ and the latent question representation $\mathbf{U}^{(t)}$, so those two models can be processed in parallel.

4.2.4 Multi-Step Iterative Reasoning with knowledge extraction. Our multi-step reasoning architecture with knowledge extraction is summarized above in Algorithm 1. Given a large-scale text corpus (as Wikipedia), the question text and candidate option set, our model returns the predicted answer choice. To narrow down the search document from the whole Wikipedia to the question relevant document set, we use the token-level retrieval followed by the semantic-level retrieval (line 3). The multi-step interaction between the selection, reformulation and entailment model can be best understood by the while loop from line 4 to line 9. The initial question $\mathbf{U}_0$ is first used to rank all the documents in the relevant documents (line 5), followed by which the *top*-1 document is sent to both the reformulation model and the entailment model. The reformulation uses the selected document to calculate the important span in the document and update the question representation $\mathbf{U}_{t+1}$ (line 6). The entailment model treats the *top*-1 selected document as the premise and the combination of question and candidate choices as the hypothesis to compute the entailment score for each choice (line 7). The updated question is sent back to the selection module (line 5). The selection module uses this updated question to re-rank the documents and the entire process is repeated for T steps. At the end of T steps, the model returns the choice with the score returned by T steps $Score_1^{(t)}, Score_2^{(t)}, ..., Score_h^{(t)}$. We update the model using log likelihood as the objective function (line 11):

$$\mathcal{L} = -\log(\frac{1}{T}\sum_{t=0}^{t=T} Score_i^{(t)}) \tag{9}$$

During inference, based on the multi-step score of each choice aggregation, the choice with the highest score is the predicted answer.

5 EXPERIMENT

We now present experiments to show the effectiveness of each component of our framework. We test the HeadQA dataset both on supervised and unsupervised settings, and further perform ablation study to evaluate the contribution from each part of the model. In the end, we present a case study to demonstrate the interpretability of our model.

5.1 Dataset

HeadQA This dataset is created from examinations, spanning the years 2013 to 2017, that are designed for obtaining specialization positions in the Spanish public healthcare areas. It contains graduate-level multi-choice questions about Medicine (MIR), Pharmacology (FIR), Psychology (PIR), Nursing (EIR), Biology (BIR), and Chemistry (QIR). The original version of this dataset is in Spanish, but it

Category	Supervised setting Train	Dev	Test	Unsupervised Setting
Biology (BIR)	452	226	454	1,132
Nursing (EIR)	384	230	455	1,069
Pharmacology (FIR)	457	225	457	1,139
Medicine (MIR)	455	231	463	1149
Psychology (PIR)	453	226	455	1134
Chemistry (QIR)	456	228	458	1142
Total	2657	1366	2742	6765

Table 1: Data Statistics Summary of HeadQA

has also been translated to English. We use the English version of this dataset. There is a total number of 6765 question-answer pairs and the questions in the Medicine category (MIR) ($\sim$ 14% among all questions) have image, which we use in question initialization. Table 1 summarizes the number of questions in each category and the data splits.

The dataset has supervised and unsupervised settings. In the supervised setting, exams from 2013 and 2014 are used for the training set, 2015 for the development set, and the rest for testing. In the unsupervised setting, we pre-trained the model on other similar tasks or datasets and test the performance of the whole dataset.

5.2 Evaluation on Reasoning Ability

In this section, we evaluate the performance of reasoning ability of **MurKe** on the HeadQA data. Since the HeadQA data is very small, we use the other related datasets to pre-train the different modules of **MurKe**. Recent studies [12, 15] have shown the benefit of fine-tuning on similar tasks or datasets for knowledge transfer. Considering the unique challenge of HeadQA, we explore the related retrieval, reading comprehension and entailment task-specific datasets for knowledge transfer. We directly adapt the pre-train weight without further training on the HeadQA dataset, which is defined as the unsupervised setting. In the supervised setting, we initialize **MurKe** with the pre-trained weight and then finetune on HeadQA.

Metrics We use Accuracy (Acc) and POINTS metric (used in the official exams): a right answer counts 3 points and a wrong one subtracts 1 point. [4]

5.2.1 Training Details. All the bi-directional GRU are with a single hidden layer (d = 200). The input of BiGRU at each token is the pre-trained BioWordVec embedding (200-dimensional) [5], which trains the word embedding using PubMed [6] and the clinical notes, and this BioWordVec covers 98% words in our dataset. Additionally, to capture the structural representation of the words, we incorporate the background knowledge in the form of graph embedding using the ConceptNet [37] knowledge base. In the end, both embeddings are concatenated to form the final word embeddings (300-dimensional).

In the unsupervised setting, the BiGRU in the retrieval model is pre-trained with the document and question encoder-decoder model. The reformulator is pre-trained using supervised learning (using the correct spans as supervision), where we use SQuAD data [32] to train this model as the pre-trained weight. The entailment model is trained using entailment task-specific dataset SciTail [20]. In the supervised setting, we first train the model by setting the number of multi-step iterative method steps (T = 1), then we train the model with different step numbers. We train the models for 50 iterations using SGD with a learning rate of 0.015 and learning rate decay of 0.05.

5.2.2 Baselines. **DrQA** [5] consists of a Document Retriever module based on bigram hashing and TF-IDF matching to return the five most relevant Wikipedia articles to a given question and a machine comprehension module, that is implemented using a multi-layer recurrent neural network trained on SQuAD [32] to find the exact span containing the correct answer. Similar to [39], the answer containing the most overlapping tokens with the selected span is considered as the correct answer for the multi-choice setting. To apply to our dataset, we calculate the similarity between selected spans and options. The option with the highest score is treated as the correct answer. **BiDAF** [33] the document retriever is the same as DrQA [5], but the Document Reader uses bi-directional attention flow mechanism and hierarchical embedding process to obtain a question-aware context representation that is used to predict the correct answer span. **DecompAttn** [20] is a textual entailment system that first forms hypothesis by appending each candidate answer option to the question. The hypothesis is then used in turn as a question to retrieve relevant sentences to be considered as the premises. The degree of a premise entailing a hypothesis is then computed as the entailment score and the answer in the hypothesis leading to the highest score is the correct answer. **DGEM** [29] is a neural attention-based entailment system that decomposes the task into sub-problems to be solved in a parallelizable manner, where the results are merged to produce the final classification output. **TFIDF Retrieval**, which is similar to the IR baselines by [8, 9], uses the DrQA [5] 's Document Retriever, which scores the relation between the queries and the articles as TF-IDF weighted bag-of-word vectors, and also takes into account word order and bi-gram counting. The predicted answer is defined as the one having the maximum score in the question for which we obtained the highest document relevance. **Retrieval + BioBERT||BERT entailment model** uses the top one document from the retrieval module and pre-trained BioBERT||BERT to get the entailment score between premise (search document) and hypothesis (question with the choice), where the top one hypothesis is the answer. **Multi-step TF-IDF retrieval (using keywords)** uses the keywords obtained from the previous document to reformulate the question and then uses TF-IDF to retrieve the new document. **Multi-step-reasoner** [10] the multi-step framework using reinforcement learning (RL) where retriever and reader (DrQA) iteratively interact with each other to get the final answer.

5.2.3 Results. **Unsupervised Setting** Table 2 shows the accuracy and POINTS scores for HeadQA. Our model **MurKe** performs best among the baselines. As can be seen, even a powerful model like BERT performs unsatisfactorily on the HeadQA dataset. The main reason is that the initial question does not contain sufficient retrievable clues to find the document containing the answer. Whereas, the multiple steps of iterative reasoning of our proposed

[4]Note that as some exams have more choices than others, there is not a direct correspondence between accuracy and POINTS (a given healthcare area might have better accuracy than another one, but worse POINTS score).
[5]https://github.com/ncbi-nlp/BioSentVec
[6]https://www.ncbi.nlm.nih.gov/pubmed/

	Acc							Point						
Models	BIR	MIR	EIR	FIR	PIR	QIR	Avg	BIR	MIR	EIR	FIR	PIR	QIR	Avg
DRQA	29.5	25.0	27.3	28.3	31.0	30.2	28.5	40.8	-0.2	20.6	29.8	54.0	47.6	32.1
BIDAF	33.4	26.2	26.8	29.9	26.8	30.3	28.9	75.6	11.0	15.8	44.4	16.6	48.6	35.3
DGEM	31.7	25.7	28.7	29.8	28.5	30.3	29.1	60.8	7.0	34.2	45.0	31.6	48.4	37.8
DECOMPATT	30.6	23.6	27.9	27.2	28.3	27.6	27.5	51.2	-13.0	27.8	20.2	30.0	23.6	23.3
TFIDF-IR	37.9	30.3	32.6	38.7	34.7	33.7	34.6	116.8	48.6	67.8	125.0	87.6	79.6	87.6
IR + BERT	29.6	31.0	33.8	33.7	30.0	33.9	32.0	41.6	55.0	76.6	79.4	45.2	82.0	63.3
IR + BioBERT	34.3	32.3	32.5	31.7	32.8	31.1	32.4	84.0	67.0	67.0	61.0	70.8	55.6	67.6
Multi-step TFIDF-IR	35.6	32.7	33.5	35.3	36.4	33.3	34.9	102.2	74.4	78.2	100.2	107.6	79.6	89.3
Multi-step Reasoner	39.7	40.1	40.2	41.3	44.0	43.0	41.7	135.0	132.6	137.4	138.6	175.6	164.4	151.3
MurKe	**45.5**	**42.4**	**42.3**	**48.0**	**44.3**	**44.3**	**44.4**	**189.4**	**158.8**	**158.8**	**209.6**	**160.6**	**173.0**	**172.3**

Table 2: Accuracy and POINTS on the HeadQA corpora (unsupervised setting)

	Acc							Point						
Models	BIR	MIR	EIR	FIR	PIR	QIR	Avg	BIR	MIR	EIR	FIR	PIR	QIR	Avg
BIDAF	36.5	26.6	27.7	29.3	28.1	34.1	30.3	104.0	14.5	18.5	39.0	29.0	83.0	48.0
DGEM	31.7	27.2	30.7	29.9	31.0	33.2	30.6	61.0	20.5	52.5	45.5	54.5	75.0	51.5
TFIDF-IR	39.8	33.3	36.4	42.2	35.7	36.0	37.2	116.8	48.6	67.8	125.0	87.6	79.6	87.6
IR + BERT	35.2	35.6	38.2	33.7	33.7	33.4	35.0	92.8	97.4	113.4	79.4	78.8	77.2	89.8
IR + BioBERT	38.0	36.4	37.8	33.7	33.6	38.9	36.4	104.6	111.0	48.6	79.4	78.0	127.6	103.0
Multi-step TFIDF-IR	41.9	38.1	36.6	39.2	40.3	39.1	39.2	155.0	118.4	99.0	129.8	138.8	129.2	128.4
Multi-step Reasoner	43.4	42.9	42.9	43.7	43.5	44.3	42.9	162.4	178.2	159.0	170.6	162.0	163.6	166.0
MurKe	**47.1**	**45.6**	**46.7**	**48.8**	**46.7**	**45.5**	**46.7**	**200.0**	**189.4**	**184.6**	**217.0**	**197.2**	**186.8**	**199.8**

Table 3: Accuracy and POINTS on the HeadQA corpora (supervised setting)

	BIR	MIR	EIR	FIR	PIR	QIR	Avg
Avg 10 best humans	627.1	592.2	515.2	575.5	602.1	529.1	477.6
Pass mark	219.0	207.0	180.0	201.0	210.0	185.0	200.3
MurKe	199.8	196.2	215.4	196.4	217.2	203.7	204.8

Table 4: Human performance on the 2016 exams (Points).

model help to reformulate the question with the missing information, which in turn facilitates in retrieving the document related to the answer and uniformly increases performance over the base model. Moreover, using different task-related datasets to pre-train each module separately is promising to achieve an acceptable performance.

Supervised Setting We show the performance of the top models on the test split corresponding to the supervised setting in Table 3. Our proposed model **MurKe** performs substantially better than the other baselines, which shows that by using the multi-step iteration it is possible to have the model better match the gold document and get a better entailment score. The other multi-step methods, like multi-step TFIDF-IR and Multi-step Reasoner perform worse than our method. This is primarily because the multi-step TFIDF-IR methods rely on statistical features like frequency of terms in the document, and fail to explicitly use information about entities that may not be frequently occurring in the document. We also find that RL approaches, Multi-step Reasoner, are slow to converge as rewards from a down-stream task are sparse and action space in information retrieval is very large. Table 4 shows humans performance. The first row is the average of the top 10 scores gotten by humans and the second row is the passing score, meaning that the examinee can pass the exam if it receives above this score. Compared to the score of the pass mark, **MurKe** passes three categories (EIR, PIR, and QIR) and the avg point is higher than the pass mark. Nevertheless, there is still a long way to beat the best performance of humans.

5.2.4 Influence of the number of Reasoning Step. As we can see from Fig. 4, without the multi-step (using 1 step), the performance is very poor 41.1. By increasing steps of interaction, the

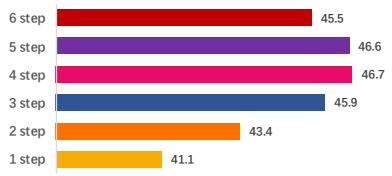

Figure 4: The accuracy with the different reasoning step.

		F1-score	Accuracy	Points
Unsupervised	w/o	42.2	42.5	158.5
	avg	42.5	42.7	163.4
Supervised	w/o	44.8	45.0	181.4
	bil	45.7	45.9	188.8
	con	45.3	45.5	186.7

Table 5: Different fusion methods on both unsupervised and supervised settings on MID data. Here avg stands for average and con for concatenate.

quality in terms of answer accuracy becomes better, which indicates that even though the correct document (containing the answer string) was not retrieved in the first step, the retriever is able to fetch relevant documents later. The performance keeps increasing with the increase of the iterative step. But when the number of steps is too high (6 steps), the performance declines, which may indicate that more noises are added with more steps. Therefore, in most cases the optimal value of T lies in a small range of values as demonstrated in [10] and it is not time-consuming to find it using the grid search strategy in practical applications. It is also unsurprising to see that when correct documents are retrieved, the performance of the entailment model also increases and it is easy to find the correct final answer.

5.2.5 Performance using multi-modality fusion. We also test the performance of the proposed model using different multi-modality fusion methods on the questions in the Medicine category (MIR)

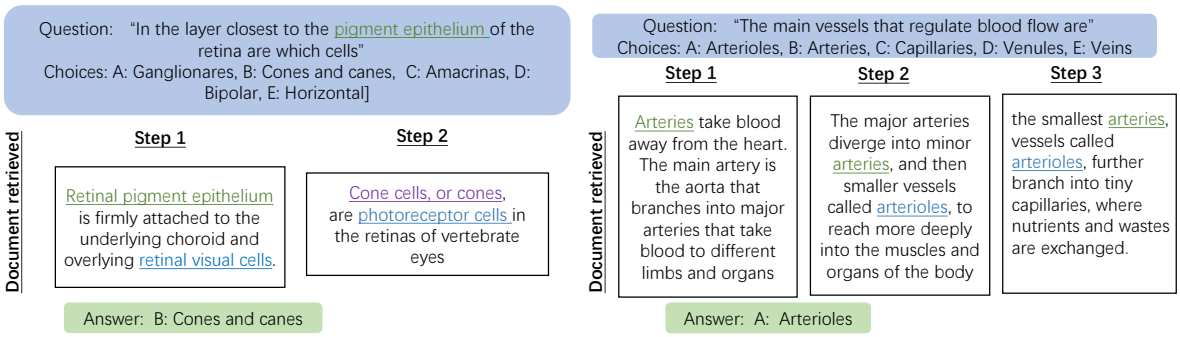

Figure 5: Examples of how multi-step reasoner iteratively modifies the question by reading context to find more relevant documents.

which have image. Among all the questions, 101 questions have image information. For the image embedding, we first loaded the pre-trained Resnet [14] model with 18 layers [7], removed its final output layer (the softmax layer), added new layers (flatten layer, dense fully connected layer of 200 units and output layer to predict probs of 10 classes). We train the weights of the pre-trained model with new layers together and then remove the output layer and extract features from the dense layer of 200 units as the embedding of each image. In the unsupervised setting, we compare without fusion method (w/o) and average the question embedding and image embedding (avg). In the supervised setting, we evaluate without fusion method (w/o), use the bi-linear model (bil) and concatenate the question embedding and image embedding and project to the pre dimension (con).

As seen from Table 5, fusing additional image information with the question representation can help improve the performance. It is not surprising that the question is related to the image, so by adding image information, it helps the model to better retrieve the document. In the supervised setting, the pre-trained weight can help the image embedding learn a projection from the image space to the text space, so it has more improvement.

5.3 Evaluation on Knowledge Extraction

In this section, we want to demonstrate the performance of the knowledge extraction which combines token-level retrieval and semantic retrieval, and indicates the necessity of using multi-step reasoning. We use the NCRF++ [45] to get the selected question key-terms and is trained on the Essential Terms dataset [8] introduced by Khashabi et al. [19]. Similarly, we use the same processing on the answer. We calculate the amount that both the question and answer appear in the same document. We find that only 21 questions (6765 in total) have one document that contains both question and answer key term, which shows the complexity of the question and the importance of using multi-step iterative method to solve this problem.

5.3.1 Influence of different supporting document scale. We want to narrow the supporting document space that not only contains the information needed by question and answer, but also will not be redundant. To know the influence of supporting document scale, instead of using threshold after the semantic-level retrieval, we rank the supporting documents and select the scale from the top 10 to top 50 to assess the performance. From Fig. 6, we can see that with the growth of supporting documents of question, the performance gets better. When the number of documents is around top 30, all methods reach the best performance. However, when the number of supporting documents goes beyond that, the performance shows a little bit drop. This may be because with the increase of documents, it leads to more deceptive documents.

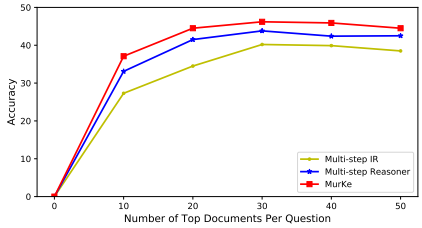

Figure 6: The accuracy with the different search document number.

5.4 Interpretable Ability of MurKe

Fig. 5 shows two instances where iterative interaction is helpful. In the left figure, the retriever is initially unable to find a document that can directly answer the question. However, it finds a document which has a different description of "visual cells", "photoreceptor cells", allowing it to find a more relevant document that directly answers the question. In the figure at the right side, the retrieved documents indicate that both "Arterioles" and "Arteries" could be the answer. Based on the fact that the smallest "arteries" is "arterioles", which reach into the muscles and organs of the body, so "arterioles" is the main vessels that regulate the blood flow. Since we aggregate (sum) the scores of entailment of each retrieved documents with choices, this leads to an increase in the score of the choice ("Arterioles") to be the predicted answer. Therefore, by using reading-answer attention in the reformulation module, our module **MurKe** can clearly show us which part in the document is highlighted as the clue regarding current question and provides the interpretability of the reasoning path.

[7] https://github.com/qubvel/classification_models
[8] https://github.com/allenai/essential-terms

6 CONCLUSIONS

In this paper, we present a system **MurKe** that answers healthcare exam questions by using knowledge extraction and multi-step reasoning. To get a relevant document for each question, **MurKe** retrieves supporting documents from a large, noisy corpus on the basis of keywords extracted from the original question and semantic retrieval. **MurKe** proposes the multi-step iterative method to solve complex healthcare QA, which uses information selected by combining iterative question reformulation and textual entailment. Our neural architecture uses a sequence of token-level attention mechanisms to extract relevant evidence from the selected documents in order to update the latent representation of the question, which shows the interpretability of the reasoning path. Through empirical results and case study, we demonstrate that our proposed system is able to outperform several strong baselines on the HeadQA dataset.

REFERENCES

[1] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*. Springer, 722–735.

[2] Lisa Bauer, Yicheng Wang, and Mohit Bansal. 2018. Commonsense for generative multi-hop question answering tasks. *arXiv preprint arXiv:1809.06309* (2018).

[3] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*. AcM, 1247–1250.

[4] Yu Cao, Meng Fang, and Dacheng Tao. 2019. BAG: Bi-directional Attention Entity Graph Convolutional Network for Multi-hop Reasoning Question Answering. *arXiv preprint arXiv:1904.04969* (2019).

[5] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051* (2017).

[6] Jifan Chen, Shih-ting Lin, and Greg Durrett. 2019. Multi-hop Question Answering via Reasoning Chains. *arXiv preprint arXiv:1910.02610* (2019).

[7] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[8] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457* (2018).

[9] Peter Clark, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Turney, and Daniel Khashabi. 2016. Combining retrieval, statistics, and inference to answer elementary science questions. In *AAAI*.

[10] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. 2019. Multi-step retriever-reader interaction for scalable open-domain question answering. *arXiv preprint arXiv:1905.05733* (2019).

[11] Nicola De Cao, Wilker Aziz, and Ivan Titov. 2018. Question answering by reasoning across documents with graph convolutional networks. *arXiv preprint arXiv:1808.09920* (2018).

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[13] Quentin Grail, Julien Perez, and Eric Gaussier. 2020. Latent Question Reformulation and Information Accumulation for Multi-Hop Machine Reading. `https://openreview.net/forum?id=S1x63TEYvr`

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*. 770–778.

[15] Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146* (2018).

[16] Phu Mon Htut, Samuel R Bowman, and Kyunghyun Cho. 2018. Training a ranking function for open-domain question answering. *arXiv preprint arXiv:1804.04264* (2018).

[17] Yichen Jiang and Mohit Bansal. 2019. Self-assembling modular networks for interpretable multi-hop reasoning. *arXiv preprint arXiv:1909.05803* (2019).

[18] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551* (2017).

[19] Daniel Khashabi, Tushar Khot, Ashish Sabharwal, and Dan Roth. 2017. Learning what is essential in questions. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. 80–89.

[20] Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *AAAI*.

[21] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[22] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *ICML*. 1378–1387.

[23] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. Biobert: pre-trained biomedical language representation model for biomedical text mining. *arXiv preprint arXiv:1901.08746* (2019).

[24] Jinhyuk Lee, Seongjun Yun, Hyunjae Kim, Miyoung Ko, and Jaewoo Kang. 2018. Ranking paragraphs for improving answer recall in open-domain question answering. *arXiv preprint arXiv:1810.00494* (2018).

[25] Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. 2018. Denoising distantly supervised open-domain question answering. In *ACL*. 1736–1745.

[26] Ye Liu, Chenwei Zhang, Xiaohui Yan, Yi Chang, and Philip S Yu. 2019. Generative question refinement with deep reinforcement learning in retrieval-based QA system. In *CIKM*. 1643–1652.

[27] Ryan Musa, Xiaoyan Wang, Achille Fokoue, Nicholas Mattei, Maria Chang, Pavan Kapanipathi, Bassem Makni, Kartik Talamadupula, and Michael Witbrock. 2018. Answering Science Exam Questions Using Query Reformulation with Background Knowledge. (2018).

[28] Rodrigo Nogueira and Kyunghyun Cho. 2017. Task-oriented query reformulation with reinforcement learning. *arXiv preprint arXiv:1704.04572* (2017).

[29] Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933* (2016).

[30] Jonas Pfeiffer, Samuel Broscheit, Rainer Gemulla, and Mathias Göschl. 2018. A neural autoencoder approach for document ranking and query refinement in pharmacogenomic information retrieval. ACL.

[31] Peng Qi, Xiaowen Lin, Leo Mehr, Zijian Wang, and Christopher D Manning. 2019. Answering complex open-domain questions through iterative query generation. *arXiv preprint arXiv:1910.07000* (2019).

[32] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250* (2016).

[33] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. In *ICLR*.

[34] Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *SIGKDD*. ACM, 1047–1055.

[35] Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in neural information processing systems*. 801–809.

[36] Linfeng Song, Zhiguo Wang, Mo Yu, Yue Zhang, Radu Florian, and Daniel Gildea. 2018. Exploring graph-structured passage representation for multi-hop reading comprehension with graph neural networks. *arXiv preprint arXiv:1809.02040* (2018).

[37] Robert Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI*.

[38] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*. 2440–2448.

[39] David Vilares and Carlos Gómez-Rodríguez. 2019. HEAD-QA: A Healthcare Dataset for Complex Reasoning. *arXiv preprint arXiv:1906.04701* (2019).

[40] Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? A quasi-synchronous grammar for QA. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. 22–32.

[41] Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang. 2018. R 3: Reinforced ranker-reader for open-domain question answering. In *AAAI*.

[42] Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *TACL* 6 (2018), 287–302.

[43] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698* (2015).

[44] Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dcn+: Mixed objective and deep residual coattention for question answering. In *ICLR*.

[45] Jie Yang and Yue Zhang. 2018. Ncrf++: An open-source neural sequence labeling toolkit. *arXiv preprint arXiv:1806.05626* (2018).

[46] Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP*. 2013–2018.

[47] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600* (2018).