Iterative Evidence Searching over Long Structured Documents for Question Answering

Anonymous ACL submission

Abstract

We propose a simple yet effective model, DOCHOPPER, for selecting evidence from long structured documents to answer complex questions. Similar to multi-hop question-answering (QA) systems, at each step, DOCHOPPER iteratively uses a query q to extract information from a document, and combines this information with q to produce the next query. However, in contrast to most previous multi-hop QA systems, DOCHOPPER is able to extract either short or long sections of the document, thus emulating a multi-step process of "navigating" through a long document to answer a question. To enable this novel behavior, DOCHOPPER does not combine document information with q by concatenating text to the text of q, but by combining a compact neural representation of q with a compact neural representation of a (potentially large) hierarchical part of the document. We evaluate DOCHOPPER on three different tasks that require reading long structured documents and finding multiple pieces of evidence, and show DOCHOPPER outperforms Transformer models for plain text input. Additionally, DOCHOPPER is efficient at inference time, being 10–250 times faster than baselines.

1 Introduction

In this work we focus on the problem of extracting evidence over long and hierarchically structured documents to answer complex questions. A long document typically contains coherent information on a certain topic, and the contents are grouped into hierarchical structures, such as sections, chapters, etc. To answer complex questions over long documents often requires navigating through different parts of the documents to find multiple pieces of information. This navigation, in turn, requires understanding high-level information about the structure of the document.

For example, consider answering questions over academic papers (Dasigi et al., 2021). To answer the question "What modules in DOCHOPPER will be finetuned in all the experiments?", one might first turn to the section titled "Model" to identify the different modules in DOCHOPPER, and then read the "Experiments" section with these modules in mind, potentially further selecting evidence from specific subsections (such as the one titled "Implementation Details"). Similar processes might be needed to answer questions concerning government policies (Sun et al., 2022) or legal documents. This type of QA tests not only the ability to understand short passages of text, but also the ability to understand the goal of questions and the structure of documents in a domain.

A common approach to solving questions that require multiple pieces of evidence is to iteratively find evidence and update the query for the next step. The update can be performed by either explicitly predicting the intermediate answers (Talmor and Berant, 2018; Sun et al., 2019) or directly appending previous evidences to the questions (Zhao et al., 2021; Qi et al., 2021; Li et al., 2020; Xiong et al., 2021). While appending retrieved evidence to a query works well on many factual QA tasks, where it is possible to answer questions with evidences that are short pieces of text, this approach is expensive if one wishes to retrieve larger pieces of text as evidences (e.g., the "Experiments" section of a paper). Another disadvantage is that appending together many small fragments of text intuitively fails to capture the relationships between them, and the structure of the document from which they were extracted.

To capture high-level structural information in a document as well as detailed information from short passages, Ainslie et al. (2020) proposed ETC, which introduced a global-local attention mechanism where embeddings of special global tokens are used to encode high-level information. [1] ETC

[1] Our DOCHOPPER system incorporates ETC as a document encoder, but other pretrained LMs will still work.

has previously performed well on multi-hop QA tasks like HotpotQA and WikiHop (Yang et al., 2018; Welbl et al., 2018) which require combining information from a small number of short passages. However, it has not been previously evaluated on tasks of the sort considered here. Our experiments show that DOCHOPPER outperforms ETC in extracting evidence for questions from long and structured documents.

DOCHOPPER proposes a novel approach to updating queries over structured documents in a multi-hop setting. DOCHOPPER iteratively attends to different parts of the document, either large parts (e.g., chapters) or small parts (e.g., sentences). This process can be viewed as either retrieving a short passage, or navigating to a part of a document. In each iteration, the query vector is updated in embedding space using the encoding of an evidence previously selected. This updating step is end-to-end differentiable and efficient. In our experiments, we show DOCHOPPER is effective on three different benchmarks involving complex queries over long and structured documents.

In particular, we evaluate DOCHOPPER on two evidence extraction tasks and one question answering (QA) task. In QASPER (Dasigi et al., 2021) and ConditionalQA (Sun et al., 2022), we evaluate DOCHOPPER's performance in extracting all evidences that are required to answer questions. In HybridQA, oracle evidence is not labeled. We instead evaluate final answer accuracy by passing the selected evidences into a simple reader model. DOCHOPPER outperforms large-document Transformer models—ETC (Ainslie et al., 2020) and Longformer (Beltagy et al., 2020)—by up to 6 points. Additionally, DOCHOPPER runs 10–250 faster than baseline models, since it makes effective use of pre-computed question-independent encodings of documents.

2 Related Work

Graph-based models have been widely used for answering multi-hop questions in factual QA (Min et al., 2020; Sun et al., 2018, 2019; Qiu et al., 2019; Fang et al., 2019). However, most of the graph-based models are grounded to entities, i.e., evidences (from knowledge bases or text corpus) are connected by entities in the graph. The graph construction step also heavily relies on many discrete features such as hyperlinks or entities predicted with external entity linkers. It's not clear how to apply these models to more general tasks if context is not entity-centric, such as questions about academic papers or government documents. Similar problems also exist in memory-augmented language models that achieved the state-of-the-art on many factual QA tasks (Guu et al., 2020; Lewis et al., 2021; Verga et al., 2020; Dhingra et al., 2020; Sun et al., 2021).

Alternatively, one can adopt a "retrieve and read" pipeline to answer multi-hop questions over long documents. Recent works proposed to extend the dense retrieval methods (Karpukhin et al., 2020) to multi-hop questions (Zhao et al., 2021; Qi et al., 2021; Li et al., 2020). However, such models retrieve one small piece of evidence at a time, lacking the ability of navigating between different parts of the documents to find relevant information at both higher and lower levels of the document-structure hierarchy. Another disadvantage of these iterative models is that they are not end-to-end differentiable. Updating the questions for the next hop requires re-encoding the concatenated tokens from the questions and previously retrieved evidences. It also makes the model inefficient because re-encoding tokens with large Transformer models is very expensive.

Besides question answering tasks, hierarchical information in documents has been successfully used in tasks such as document classification (Yang et al., 2016; Chang et al., 2019), summarization (Gidiotis and Tsoumakas, 2020; Xiao and Carenini, 2019; Zhang et al., 2019), sentiment analysis (Ruder et al., 2016), text segmentation (Koshorek et al., 2018), etc. It is worth mentioning that ETC (Ainslie et al., 2020) was also used on a key-phrase extraction task on web pages using structured DOM trees. However, none of these models can be easily adapted to answering complex questions over long documents.

3 Model

In this section, we discuss the iterative process of extracting evidence from long and structured documents. The iterative process is performed over a pre-computed document index that contains embeddings at different hierarchical levels. To start with, we first introduce strategies to compute embeddings for parts of a document to build an index for a document. Then, we present the iterative process that operates over document index to extract evidence. Depending on hierarchical level of the

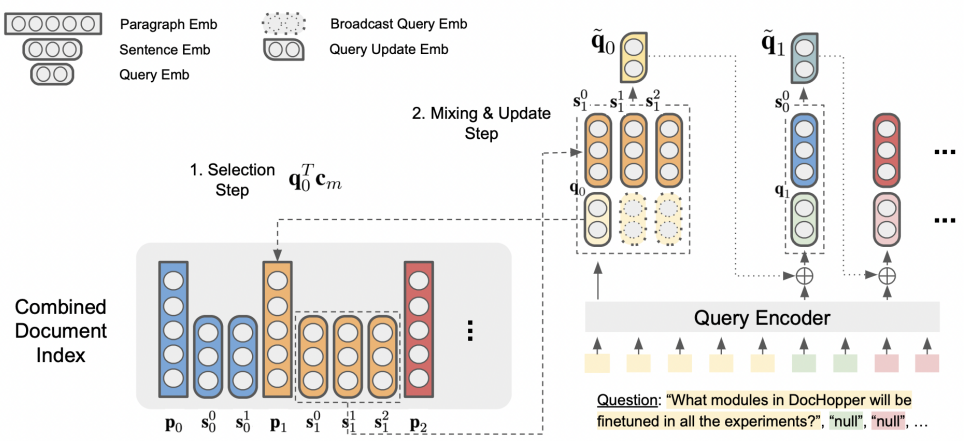

Figure 1: DOCHOPPER Overview. For a structured document consisting of sentences and paragraphs, during the iterative selection process, DOCHOPPER selects a paragraph or a sentence from a combined document index that contains both paragraph embeddings and sentence embeddings. Selected information will be mixed with the query vector and in turn update the query for the next hop. Different update strategies are applied if sentences or paragraphs are selected previously.

evidence selected, a different query update strategy will be applied.

3.1 Input

A long document usually contains multiple levels of hierarchy, e.g. sections, sub-sections, paragraphs, sentences, etc. For simplicity, we only consider two levels of hierarchy in this paper: *paragraph-level* and *sentence-level*. A *sentence* is the lowest granularity that can be selected, while a *paragraph* is an abstraction of a collection of sentences, which can be used to represent sections or other levels in the hierarchy, depending on the application. Formally, let $d = \{p_0, \ldots, p_{|d|}\} \in D$ be a document in the corpus D that contains a sequence of paragraphs p_j, and let a paragraph $p_j = \{s_j^0, \ldots, s_j^{|p_j|}\}$ contain a sequence of sentences. A sentence s_j^i will be encoded into a fixed length vector $\mathbf{s}_j^i \in \mathbb{R}^d$.

3.2 Document Index in Embedding Space

Sentence Embeddings A sentence s_j^i has the lowest granularity that can be selected as evidence. We learn a Transformer model to encode sentences s_j^i into vectors $\mathbf{s}_j^i$.

$$\mathbf{s}_j^i = \text{Transformer}_{\text{sent}}(s_j^i) \quad (1)$$

Paragraph Embeddings Paragraph embeddings are derived from sentence embeddings $\mathbf{s}_j^i$ and dependent on queries $\mathbf{q}_t$, the embedding of the t'th hop of the question. We will discuss methods to obtain query embeddings $\mathbf{q}_t$ later in §3.3. A paragraph embedding $\mathbf{p}_j$ is the weighted sum of sentence embeddings $\mathbf{s}_j^i$ in paragraph p_j, where α_i is the attention weights of the query vector $\mathbf{q}_t$ to the sentence embedding $\mathbf{s}_j^i$.

$$\mathbf{p}_j = \sum_i \alpha_i \, \mathbf{s}_j^i, \quad \alpha_i = \text{softmax}(\mathbf{q}_t^T \mathbf{s}_j^i) \quad (2)$$

The paragraph embeddings $\mathbf{p}_j$ are thus dependent on the query, but do not require jointly encoding tokens from queries and context, as in many BERT-style reading comprehension models. Computing paragraph embeddings with Eq.2 is hence very efficient.

Combined Document Index We put the sentence embeddings and paragraph embeddings of document d into a combined document index, so the model has the flexibility to decide which sentence or paragraph to attend to. Different update rules will be applied according to whether sentences or paragraphs are attended to.

To construct the embedding table, we iterate through all paragraphs in a document and apply the sentence encoder to compute sentence and paragraph embeddings. Sentence and paragraph embeddings from all paragraphs are then concatenated to form a combined embedding table. We denote the combined embedding table for document d as $\mathbf{C}_d = \{\mathbf{p}_0, \mathbf{s}_0^0, \ldots, \mathbf{s}_0^{|p_0|}, \mathbf{p}_1, \mathbf{s}_1^0, \ldots, \mathbf{s}_1^{|p_1|}, \ldots\}$. Let $\mathbf{c}_m$ be the embedding of the m'th entry from $\mathbf{C}_d$; we emphasize that $\mathbf{c}_m$ can represent either a sentence or a paragraph embedding.

Pretrained Sentence Encoder We use ETC (Ainslie et al., 2020) as our sentence encoder, as it is pretrained to produce sentence-level embeddings. Different from vanilla Transformer models, e.g. BERT (Devlin et al., 2019), ETC introduces

an global-local attention mechanism. ETC assigns to each sentence a special *global token* that only attends to local tokens in the sentence, and its embedding is trained to summarize the information of local tokens in the sentence. A global token also attends the global tokens of other sentences in the input. ETC additionally adopts Contrastive Predictive Coding (CPC) (Oord et al., 2018) to train the embedding of global tokens to make them aware of other sentences in the context. We use the embeddings of global tokens in ETC as sentences embeddings.

Specifically, instead of encoding one sentence at a time, we run ETC over multiple contiguous sentences (usually a paragraph) to improve encoding efficiency, $p_j = \{s_j^0, \ldots, s_j^{|p_j|}\}$. ETC's output includes vectors $\mathbf{s}_j^0, \ldots, \mathbf{s}_j^{|p_j|}$, where each $\mathbf{s}_j^i \in \mathbb{R}^d$ represents the embedding of a sentence s_j^i.

$$\mathbf{s}_0^i, \ldots, \mathbf{s}_{|p_i|}^i = \text{ETC}(\{s_0^i, \ldots, s_{|p_i|}^i\}) \ \in \mathbb{R}^{|p_i| \times d}$$

Finetuning Sentence Encoder A pretrained ETC model can be finetuned to specific domains. While finetuning ETC's sentence encoder generally improves performance of our model, we find the pretrained ETC produces reasonably good sentence embeddings without finetuning, and using pretrained ETC allows faster training.

3.3 Query Embeddings

Many questions, especially ones answered by professional documents that are long and structured, require navigating through different parts of documents to find multiple pieces of evidence. We consider it as an iterative search process over the precomputed document index that has been discussed above. The search process is performed in embedding space.

Different from the multi-hop questions that have been studied in past work (Sun et al., 2018; Qiu et al., 2019; Min et al., 2020; Chen et al., 2020), e.g. "*Which gulf is north of the Somalian city with 550,000 residents*", we focus on questions that requires information from multiple parts of a document that are *hierarchically related*. For example, a question that asks "am I eligible for this benefit" may require first navigating to a section that describes the requirements of the benefit based on the user's scenario, and then check whether all requirements have been satisfied. This searching process is inherently multi-hop and requires combining both contextual and hierarchical information.

Assume that a question is k-hop, where k is a hyper-parameter.To generate k different query vectors, one for each hop, we add $k-1$ dummy questions q_{null} to form a question paragraph $q_p = \{q_0, q_{\text{null}}, \ldots, q_{\text{null}}\}$. The question paragraph is passed into a query encoder to compute query embeddings.

$$\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_{k-1} = \text{Encoder}_q(\{q_0, q_{\text{null}}, \ldots, q_{\text{null}}\}) \quad (3)$$

Again, we use pretrained ETC as our query encoder in this project. The global-to-local attention mask of ETC is modified to allow the global token of the dummy question to attend to tokens in the question q_0. With this modification, query embeddings for q_0 and q_{null} can attend to different parts of the question. ETC is always finetuned as query encoder. Query vectors $\mathbf{q}_t$ will not be directly used to select evidence at the t'th step, but instead will be updated using previously selected information before selecting next evidence.

Query vectors can either select paragraphs or sentences from documents. The selection process is performed over the combined document index that contains both sentence and paragraph embeddings. The selection process will be discussed in the next section.

3.4 Iterative Evidence Selection

With the query embedding $\mathbf{q}_t$ at step t and context embeddings $\mathbf{C}_d$ discussed above, we now introduce the proposed iterative evidence selection algorithm in DOCHOPPER.

Selection Step At each iteration, DOCHOPPER computes inner product scores between the query vector $\mathbf{q}_t$ and embeddings $\mathbf{c}_m$ in $\mathbf{C}_d$, and returns the entry $\hat{c}$ with the largest score, which is usually referred as hard attention. (As we will see $\hat{c}$ is not directly used for computation, but it is helpful in explaining the selection step).

$$\hat{c} = \text{argmax}_{c_m}(\mathbf{q}_t^T \mathbf{c}_m)$$

Note that the selected entry can be either a paragraph p_j or a sentence s_j^i because the document index $\mathbf{C}_d$ contains both sentence and paragraph embeddings.

Update Step Many multihop models update a question by appending retrieved text to the text of the question. In contrast, DOCHOPPER numerically combines the embedding of the selected entry $\hat{c}$

with the embedding of the query vector $\mathbf{q}_t$, a process we call "mixing". Since the combined embedding table $\mathbf{C}_d$ contains both sentence and paragraph embeddings, the selected entry $\hat{c}$ can represent either a sentence or a paragraph. The two cases will be considered separately. If $\hat{c}$ is a sentence, i.e. $\hat{c} = s_j^i$, DOCHOPPER computes the mixed embedding as

$$\tilde{\mathbf{q}}_t = \mathbf{W}_q^T \, [\mathbf{q}_t; \mathbf{s}_j^i] \tag{4}$$

where $[\mathbf{q}_t; \mathbf{s}_j^i]$ is the concatenation of two vectors $\mathbf{q}_t$ and $\mathbf{s}_j^i$. The mixed vector is then used to update the query to form $\mathbf{q}_{t+1}$ as shown in Eq. 5. Intuitively, $\tilde{\mathbf{q}}_t$ is the residual from the previous step. Adding the residual embedding encourages the model to attend to information that is not fully satisfied from previous steps.

$$\mathbf{q}_{t+1} \leftarrow \mathbf{q}_{t+1} + \tilde{\mathbf{q}}_t \tag{5}$$

If $\hat{c}$ is a paragraph, i.e., $\hat{c} = p_j$, a more complex mixing process is used. DOCHOPPER first looks up the sentences in p_j, i.e. the vectors $\{s_j^0, \ldots, s_j^{|p_j|}\}$. The following three steps are then used to compute the update vector $\tilde{\mathbf{q}}_t$. (1) DOCHOPPER computes the importance weights of the query vector $\mathbf{q}_t$ to the embeddings of associated sentences $\{s_j^0, \ldots, s_j^{|p_j|}\}$ that measures the relevance scores between the query vector and the sentences. This importance weight is the same as the weight α_i in Eq.2 that is used to compute the paragraph embeddings. In the implementation, we also re-use the value of α_i if it has been computed for the query-dependent paragraph embeddings. (2) The query vector $\mathbf{q}_t$ is combined with every sentence in paragraph. In particular, $\mathbf{q}_t$ is multiplied with weight α_i and appended to the i-th sentence embedding $\mathbf{s}_j^i$, where the α_i's indicate relevance. The result is then linearly projected to form a vector $\mathbf{k}_i$:

$$\mathbf{k}_j = \mathbf{W}_q^T [\alpha_i \, \mathbf{q}_t; \mathbf{s}_j^i] \tag{6}$$

Then (3) the vectors $\mathbf{k}_i$ are summed with the weight β_i, where β_i is the attention weight of a learned vector $\mathbf{v}$ to the concatenated vector $\mathbf{k}_i$. The learned vector $\mathbf{v}$ weights the importance of sentences from the selected paragraph after comparing them with the query vector and decides what information to pass to the next step of selection.

$$\tilde{\mathbf{q}}_t = \sum_i \beta_i \, \mathbf{k}_i, \quad \beta_i = \text{softmax}(\mathbf{v}^T \mathbf{k}_i) \tag{7}$$

It is not hard to see that computing the mixed embedding in Eq. 7 for the case that a paragraph is selected is essentially the same as in Eq. 4 if the selected paragraph p_i only contains one sentence, i.e. $\alpha_i = 1$ and $\beta_i = 1$ if $|p_j| = 1$; hence the same logic can be used regardless of whether $\hat{c}$ is a sentence or a paragraph.

Loss Function Attention is supervised if (distantly) supervised labels are available in the dataset. $\mathbf{q}_t^T \mathbf{c}_m$ is the inner product score between the query vector $\mathbf{q}_t$ and a context embedding $\mathbf{c}_m$. $\mathbb{I}_{c_m}$ is an indicator function that equals to 1 iff the label of c_m is positive.

$$l_t = \text{cross\_entropy}(\text{softmax}(\mathbf{q}_t^T \mathbf{c}_m), \mathbb{I}_{c_m})$$

The loss function is computed at the final step, and possibly at intermediate steps if labels are available. Supervision labels are sometimes distantly constructed. For example, in the extractive QA task, a positive candidate is the sentence or paragraph that contains the answer span (see §4).

3.5 Evidence Prediction

After all iterations, scores at all iterative steps are summed to compute a final score which is used to make prediction. The score for sentence s_j^i is computed as

$$\text{score}(s_j^i) = \sum_t \lambda_t \cdot (\mathbf{q}_t^T \mathbf{s}_j^i + \mathbf{q}_t^T \mathbf{p}_j) \tag{8}$$

where $\mathbf{q}_1^T \mathbf{s}_j^i$ and $\mathbf{q}_0^T \mathbf{p}_j$ are the scores of sentence s_j^i and paragraph p_j that it belongs to. λ_t are hyperparameters tuned for different datasets. We often set $\lambda_0 = 1$ and tune the rest of λ_t's.

3.6 Runtime Efficiency

DOCHOPPER is very efficient at runtime thanks to the query-agnostic sentence embeddings that can be pre-computed (§3.2) at inference time. Different from previous reading comprehension models that jointly encode questions and context (Beltagy et al., 2020; Ainslie et al., 2020), DOCHOPPER encode question embeddings and context embeddings independently. At inference time, DOCHOPPER directly select from document index that contains precomputed context embeddings, significantly reducing the computation cost compared to cross-attention models that jointly encode questions and context.

4 Experiments

We evaluate DOCHOPPER on two evidence extraction tasks, QASPER (Dasigi et al., 2021) and ConditionalQA (Sun et al., 2022). QASPER contains questions about academic papers. ConditionalQA contains questions about public policies described on government websites. Documents in both datasets are long and structured, and answering the questions requires navigating through entire documents to find relevant information. Both datasets provide labels for the evidence that is required to find answers, which we use to evaluate the evidence extracted by DOCHOPPER.

In addition to QASPER and ConditionalQA, we additionally evaluate DOCHOPPER on a variant of HybridQA (Chen et al., 2020). HybridQA contains multihop data that requires using both text and tabular data. Here, following (Chen et al., 2021), we consider an alternative setting where tables are preprocessed into structured documents—i.e. cells in tables are converted into sentences and sentences for cells in the same row are then merged into a paragraph. Please see §4.1 for more information. Since evidence is not labeled in HybridQA, we run a simple reader on the extracted evidence and report numbers in final answer accuracy (in EM/F1).[2]

4.1 Datasets

QASPER (Dasigi et al., 2021) (CC BY 4.0 License) is a QA dataset constructed from NLP papers. Questions are asked without reading the full paper and thus usually requires combining multiple pieces of information to obtain final answers. As it is mentioned in Dasigi et al. (2021), 55.5% of the questions have multi-paragraph evidence. Documents in the QASPER dataset are highly structured, i.e. contents are structured into sections, subsections, etc. We treat each subsection as a paragraph and prepend the section and subsection titles to the beginning of the subsection.

ConditionalQA (Sun et al., 2022) (CC BY-SA 4.0 License) contains questions on public policies that are asked over documents posted on government websites. Similar to QASPER, documents in ConditionalQA are also highly structured, with information structured in sections, subsections, listed items, tables etc. Documents in ConditionalQA are presented in HTML format. We treat HTML elements at the leaf of the DOM tree as sentences and group sentences that share the same parents as paragraphs. The ConditionalQA dataset also provides a list of evidence which we use to evaluate extraction results by DOCHOPPER.[3]

HybridQA (Chen et al., 2020) (CC BY 4.0 License) is a dataset that requires jointly using information from tables and hyperlinked text from cells to find the answers. In this experiment, we consider HybridQA in a long document QA setting, where tables are converted to structured documents with paragraphs and sentences. Annotated evidence is not provided in HybridQA, so we evaluate DOCHOPPER on the final predicted answers.

A row in the table describes attributes of an instance, for example, a person or an event. Attributes are organized by columns. For example, the table of Medalist of Sweden in 1932,[4] contains a row "[Medal:] *Gold*; [Name:] *Rudolf Svensson*; [Sport:] *Wrestling (Greco-Roman)*; [Event:] *Men's Heavyweight*". Text in the square brackets are the headers of the table. The medal winner "*Rudolf Svensson*" and the event "*Wrestling (Greco-Roman)*" are hyperlinked to the first paragraph of their Wikipedia pages. A question asks "*What was the nickname of the gold medal winner in the men's heavyweight greco-roman wrestling event of the 1932 Summer Olympics?*" requires the model to first locate the correct row in the table, and find the answer from other cells in the row or their hyperlinked text.

We convert a table with hyperlinked text into a long document. Each row in the table is considered a paragraph by concatenating the column header, cell text, and hyperlinked text if any. The column name and cell text are each treated as one sentence. Hyperlinked text is also split into sentences. In the example above, the row becomes "*Medal. Gold. Name. Rudolf Svensson. Johan Rudolf Svensson (27 March 1899 – 4 December 1978) was a Swedish wrestler. He competed ...*". The average length of the documents is 9345.5 tokens.

4.2 Implementation Details

For QASPER and ConditionalQA, we construct distant labels to supervise DOCHOPPER to first select paragraphs, and then select sentences. Note that we do not require that sentences selected in

[2] All datasets are released for research purposes.

[3] Examples in ConditionalQA have *conditional answers*, i.e. answers are only correct under certain conditions. We will leave this new task for future work and focus on evidence extraction in this paper.

[4] https://en.wikipedia.org/wiki/Sweden_at_the_1932_Summer_Olympics

	QASPER		ConditionalQA	
	Dev	Test	Dev	example / sec
Retrieval + ETC	22.4	27.9	21.1	8.3 / s
Sequential ETC	22.8	28.7	23.7	0.7 / s
LED (Longformer)	23.9	29.6	**29.4**	0.5 / s
FiD	24.9	32.3	24.6	1.3 / s
DOCHOPPER	**25.8**	**33.1**	28.8	124.8 / s
(sentence only)	24.4	29.8	27.3	–
(single-hop)	22.8	28.0	26.4	–
(w/o query update)	25.1	31.8	26.5	–

Table 1: F1 results in evidence selection. Results of baselines are obtained by running open-sourced codes.

the second hop must be from the previously selected paragraphs. Final scores for prediction is computed as described in Eq. 8. We set $\lambda_1 = 0.5$ for QASPER and $\lambda_1 = 1.2$ for ConditionalQA.

In HybridQA, we additionally use paragraph-level sparse features to improve accuracy, similar to the baseline model by Chen et al. (2020). The function $\text{sparse}(q_0, p_i)$ computes the length of longest common substrings in the question q_0 and the paragraph p_i. Sparse features are only used at the end of retrieval, not at any intermediate steps. For HybridQA, we set $\lambda_1 = 1.5$ and $\gamma = 3.0$.

$$\text{score}(s^i_j) \leftarrow \text{score}(s^i_j) + \gamma \cdot \text{sparse}(q_0, p_i) \quad (9)$$

Since oracle evidence is not provided in HybridQA, we consider sentences that contain answers as evidence. Selected evidence is then passed to a BERT-based model to extract final answers. We finetune BERT-large to serve as our reader. Experimental results are presented in Table 2.

4.3 Baselines

We compare DOCHOPPER with strong baselines for long input—LED (Longformer) (Dasigi et al., 2021), FiD (Izacard and Grave, 2020), and variations of ETC (Ainslie et al., 2020)—to show the efficacy of DOCHOPPER. LED is an encoder-decoder model that builds on Longformer (Beltagy et al., 2020). Fusion-in-Decoder (FiD) is based on T5 but uses the fusion-in-decoder strategy to reduce memory usage for longer inputs. We also experiment with directly reading the documents with a Transformer-based reader ETC (Ainslie et al., 2020): though it can't fit the entire document into its input, it is still one of the best models for reading long sequences (up to 4096 tokens). To handle longer documents, we adopt the sequential reading strategy: the model reads the document paragraph by paragraph, and picks the most confident prediction as the answer. We also report the numbers of a "retrieve and read" pipeline with a dense retriever (DPR-like) and a finetuned ETC reader. The numbers are shown in Table 1. Runtime is measured as examples per second with a batch size of 1.

For HybridQA, we additionally compared to QA models that are specifically designed for tabular data, e.g. HYBRIDER (Chen et al., 2020), MATE (Eisenschlos et al., 2021), and MITQA (Kumar et al., 2021). Again, we focus on the task of extracting evidences from long and structured documents and thus convert tables in HybridQA into plain text. Some tabular information, such as cell and column structures, has been removed in this conversion process. Although DOCHOPPER is not directly comparable models specialized for tables, we also present numbers for sevreal such models—HYBRIDER (Chen et al., 2020), MATE (Eisenschlos et al., 2021), and MITQA (Kumar et al., 2021)–in Table 2 for completeness.

4.4 Results and Analysis

On QASPER, DOCHOPPER outperforms the baselines by 1-5%, and runs 10-250 times faster. On ConditionalQA, DOCHOPPER's retrieval performance is slightly worse than LED (0.6%), but it is 250 times faster. We additionally performed more ablation experiments with DOCHOPPER. The query update (see the row w/o query update) in Eq. 5 is important, causing 0.7% and 2.3% difference in performance on both datasets. We also ablated the model by using one step of attention to select the most relevant sentence from the document (single-hop), and note again that performance drops noticeably. Adding one more step of attention, while only attending to sentences (sentence-only in the table), leads to some improvement, but is still worse than attending at both paragraph and sentence levels.

HybridQA is evaluated on answer accuracy be-

	HybridQA Dev	HybridQA Test	example / sec
Retrieval + ETC	37.0 / 43.5	34.1 / 40.3	8.3 / s
Sequential ETC	39.4 / 44.8	37.0 / 43.0	0.5 / s
Longformer	45.8 / 53.5	43.4 / 49.7	0.5 / s
DOCHOPPER	**47.7 / 55.0**	**46.3 / 53.3**	74.6 / s
HYBRIDER	44.0 / 50.7	43.8 / 50.6	–
DOCHOPPER(w/ cell)	53.1 / 61.4	– / –	–
MATE	63.4 / 71.0	62.8 / 70.2	–
MITQA	**65.5 / 72.7**	**64.3 / 71.9**	–
(w/o sparse)	44.4 / 51.2	– / –	–
(w/o query update)	44.2 / 50.9	– / –	–
(sentence-only)	36.7 / 43.7	– / –	–
(single-hop)	27.8 / 34.1	– / –	–

Table 2: EM/F1 performance of answer spans on HybridQA. Results of baseline models are obtained by running open-sourced codes.

	HybridQA
DOCHOPPER	**56.5**
(w/o sparse)	53.3
(w/o query update)	51.8
(sentence-only)	46.4
(single-hop)	34.2

Table 3: Hits@1 accuracy on distantly labeled evidence, i.e. sentences that contain answers (on dev set).

cause oracle evidence is not provided. DOCHOPPER with a simple BERT reader outperforms a few Transformer baselines for long input, e.g. ETC (Ainslie et al., 2020) and Longformer (Beltagy et al., 2020). DOCHOPPER outperforms baselines by 1.5-3.6 points. Again, DOCHOPPER with a BERT reader overall performs 9-150 times faster than the Transformer baselines.

Comparing to QA models specifically designed for tabular data, i.e. HYBRIDER (Chen et al., 2020), MATE (Eisenschlos et al., 2021), and MITQA (Kumar et al., 2021), DOCHOPPER does not perform, as well because our process of converting tables to DOCHOPPER's hierarchical-document input format loses information about cells and columns, that are needed to fully understand tabular data. We also extended DOCHOPPER to allow it to return cells that contain selected sentences and pass the cells (instead of single sentences) to the underlying extractive QA model (labeled "w/ cell" in the table). This improves the performance by 5.4 points. We note that the baselines for tabular data, e.g. MATE, are optimized for this task in other ways: e.g., they restrict the length of text in cells to a limited number of sentences, only the use the top-k sentences from the hyperlinked text, and restrict the total length of tables to 2048 tokens. DOCHOPPER does not impose these restrictions, and can applied to more general tasks (as shown in the other experiments).

To show the efficacy of the proposed iterative evidence extraction method, we present the Hits@1 accuracy of selecting distantly labeled evidence, i.e. cells (converted to sentences) that contain correct answers. These results are shown in Table 3, along with ablated results.

5 Conclusion

We consider on the problem of extracting evidence for complex questions over long and structured documents. Like multi-hop open QA tasks, this problem requires not only conventional "machine reading" abilities, but the ability to extract relevant information and refine queries based on retrieved information. Additionally, it requires the ability to navigate through a document, by understanding the relationship between sections of the document and parts of the question. Unlike most prior multi-hop QA models, queries in DOCHOPPER are updated in embedding space, rather than by appending to a discrete representation of question text. This approach is end-to-end differentiable and very fast. Experiments also demonstrate that this use of iterative searching can significantly improve the performance in selecting evidence from long and structured documents: in fact, the DOCHOPPER model outperforms Transformer baselines by 3–5%, while also being 10-250 times faster. However, DOCHOPPER's performance is still limited (e.g. only 28.8 in evidence F1) and thus needs substantial improvement for real world applications.

References

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. Etc: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 268–284.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Ming-Wei Chang, Kristina Toutanova, Kenton Lee, and Jacob Devlin. 2019. Language model pre-training for hierarchical document representations. *arXiv preprint arXiv:1901.09128*.

Wenhu Chen, Ming-Wei Chang, Eva Schlinger, William Yang Wang, and William W. Cohen. 2021. Open question answering over tables and text. In *International Conference on Learning Representations*.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Wang. 2020. Hybridqa: A dataset of multi-hop question answering over tabular and textual data. *Findings of EMNLP 2020*.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. *arXiv preprint arXiv:2105.03011*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Bhuwan Dhingra, Manzil Zaheer, Vidhisha Balachandran, Graham Neubig, Ruslan Salakhutdinov, and William W Cohen. 2020. Differentiable reasoning over a virtual knowledge base. *arXiv preprint arXiv:2002.10640*.

Julian Martin Eisenschlos, Maharshi Gor, Thomas Müller, and William W. Cohen. 2021. Mate: Multi-view attention for table transformer efficiency.

Yuwei Fang, Siqi Sun, Zhe Gan, Rohit Pillai, Shuohang Wang, and Jingjing Liu. 2019. Hierarchical graph network for multi-hop question answering. *arXiv preprint arXiv:1911.03631*.

Yifan Gao, Chien-Sheng Wu, Jingjing Li, Shafiq Joty, Steven CH Hoi, Caiming Xiong, Irwin King, and Michael R Lyu. 2020. Discern: Discourse-aware entailment reasoning network for conversational machine reading. *arXiv preprint arXiv:2010.01838*.

Alexios Gidiotis and Grigorios Tsoumakas. 2020. A divide-and-conquer approach to the summarization of long documents. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:3029–3040.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*.

Gautier Izacard and Edouard Grave. 2020. Leveraging passage retrieval with generative models for open domain question answering. *CoRR*, abs/2007.01282.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Omri Koshorek, Adir Cohen, Noam Mor, Michael Rotman, and Jonathan Berant. 2018. Text segmentation as a supervised learning task. *arXiv preprint arXiv:1803.09337*.

Vishwajeet Kumar, Saneem Chemmengath, Yash Gupta, Jaydeep Sen, Samarth Bharadwaj, and Soumen Chakrabarti. 2021. Multi-instance training for question answering across table and linked text.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks.

Shaobo Li, Xiaoguang Li, Lifeng Shang, Xin Jiang, Qun Liu, Chengjie Sun, Zhenzhou Ji, and Bingquan Liu. 2020. Hopretriever: Retrieve hops over wikipedia to answer complex questions.

Sewon Min, Danqi Chen, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2020. Knowledge guided text retrieval and reading for open domain question answering.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Peng Qi, Haejun Lee, Oghenetegiri "TG" Sido, and Christopher D. Manning. 2021. Retrieve, read, rerank, then iterate: Answering open-domain questions of varying reasoning steps from text.

Lin Qiu, Yunxuan Xiao, Yanru Qu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. 2019. Dynamically fused graph network for multi-hop reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6140–6150.

Sebastian Ruder, Parsa Ghaffari, and John G Breslin. 2016. A hierarchical model of reviews for aspect-based sentiment analysis. *arXiv preprint arXiv:1609.02745*.

Marzieh Saeidi, Max Bartolo, Patrick Lewis, Sameer Singh, Tim Rocktäschel, Mike Sheldon, Guillaume Bouchard, and Sebastian Riedel. 2018. Interpretation of natural language rules in conversational machine reading. *arXiv preprint arXiv:1809.01494*.

Haitian Sun, Tania Bedrax-Weiss, and William W. Cohen. 2019. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text.

Haitian Sun, William Cohen, and Ruslan Salakhutdinov. 2022. ConditionalQA: A complex reading comprehension dataset with conditional answers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3627–3637, Dublin, Ireland. Association for Computational Linguistics.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. *arXiv preprint arXiv:1809.00782*.

Haitian Sun, Pat Verga, Bhuwan Dhingra, Ruslan Salakhutdinov, and William W Cohen. 2021. Reasoning over virtual knowledge bases with open predicate relations. *arXiv preprint arXiv:2102.07043*.

A. Talmor and J. Berant. 2018. The web as a knowledge-base for answering complex questions. In *North American Association for Computational Linguistics (NAACL)*.

Pat Verga, Haitian Sun, Livio Baldini Soares, and William W Cohen. 2020. Facts as experts: Adaptable and interpretable neural memory over symbolic knowledge. *arXiv preprint arXiv:2007.00849*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302.

Wen Xiao and Giuseppe Carenini. 2019. Extractive summarization of long documents by combining global and local context. *arXiv preprint arXiv:1909.08089*.

Wenhan Xiong, Xiang Lorraine Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Wen tau Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oğuz. 2021. Answering complex open-domain questions with multi-hop dense retrieval.

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. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489.

Xingxing Zhang, Furu Wei, and Ming Zhou. 2019. Hibert: Document level pre-training of hierarchical bidirectional transformers for document summarization.

Chen Zhao, Chenyan Xiong, Jordan Boyd-Graber, and Hal Daumé III au2. 2021. Multi-step reasoning over unstructured text with beam dense retrieval.

A Appendix

A.1 Dataset Details

HybridQA (Chen et al., 2020) is a dataset that requires jointly using information from tables and text hyperlinked from table cells to find the answers of multi-hop questions. A row in the table describes attributes of an instance, for example, a person or an event. Attributes are organized by columns. For example, the table of Medalist of Sweden in 1932,[5] contains a row "[Medal:] *Gold*; [Name:] *Rudolf Svensson*; [Sport:] *Wrestling (Greco-Roman)*; [Event:] *Men's Heavyweight*". Text in the square brackets are the headers of the table. The medal winner "*Rudolf Svensson*" and the event "*Wrestling (Greco-Roman)*" are hyperlinked to the first paragraph of their Wikipedia pages. A question asks "*What was the nickname of the gold medal winner in the men 's heavyweight greco-roman wrestling event of the 1932 Summer Olympics?*" requires the model to first locate the correct row in the table, and find the answer from other cells in the row or their hyperlinked text.

To apply our model on the HybridQA dataset, we first convert a table with hyperlinked text into a long document. Each row in the table is considered a paragraph by concatenating the column header, cell text, and hyperlinked text if any. The column name and cell text are each treated as one sentence. Hyperlinked text is also split into sentences. In the example above, the row becomes "*Medal. Gold. Name. Rudolf Svensson. Johan Rudolf Svensson (27 March 1899 – 4 December 1978) was a Swedish wrestler. He competed ...*". The average length of the documents is 9345.5.

QASPER (Dasigi et al., 2021) is a QA dataset constructed from NLP papers. They hired graduate students to read the papers and ask questions. A different group of students are hired to answer the questions. For example, a question asks "What are the baseline models used in this paper?". The answers are {"*BERT*", "*RoBERTa*"}. The dataset contains a mixture of extractive, abstractive, and yes/no questions. We focus on the subset of extractive questions (51.8% of the datasets) in this paper. Some questions in the dataset are answerable with a single-hop. However, as suggested in the original paper, 55.5% of the questions have multi-paragraph evidence, and thus aggregating multiple pieces of information should improve the accuracy. Answers

[5] https://en.wikipedia.org/wiki/Sweden_at_the_1932_Summer_Olympics

in the QASPER dataset are longer, with an average of 14.4 tokens. We treat each subsection as a paragraph and prepend the section title and subsection title to the beginning of the subsection.

ShARC (Saeidi et al., 2018) is a conversational QA dataset for discourse entailment reasoning. Questions in ShARC are about government policy crawled from government websites. Users engage with a machine to check if they qualify for some benefits. A question in the dataset starts with a initial question, e.g. "*Can I get standard deduction for my federal tax return?*", with a user scenario, e.g. "*I lived in the US for 5 years with a student visa*", and a few followup questions and answers through the interaction between the machine and users, e.g. "Bot: *Are you a resident alien for tax purpose?* User: *No*". The model reviews the conversation and predicts one of the three labels: "Yes", "No", or "Irrelevant". If the model think there's not enough information to make the prediction, it should predict a fourth label "Inquire".

Besides the conversation, each example in the ShARC dataset provides a snippet that the conversation is originated from. A snippet is a short paragraph that the conversation is created from, e.g. "*Certain taxpayers aren't entitled to the standard deduction: (1) A married individual filing as married... (2) An individual ...*". Since the snippets are usually short, with an average of 54.7 tokens, previous models, e.g. DISCERN (Gao et al., 2020), concatenate the snippet and the conversation, and jointly encode them with Transformer-based models, e.g. BERT or RoBERTa. Here we consider instead a more challeging long-document setting, in which the snippet is not known, and the model must also locate the snippet from the document. We crawl the web pages with the provided URL. The pages contain 737.1 tokens on average, 13.5 times longer than the original snippets, and the longest page contains 3927 tokens. We name this new variant ShARC-Long.

A.2 Dataset Statistics

Dataset statistics are shown in Table 4.

	Train	Dev	Test
QASPER	2593	1005	1451
ConditionalQA	2338	285	804
HybridQA	62682	3466	3463

Table 4: Dataset statistics.

A.3 Implementation Details for ShARC-Long

Changes to Context Representations Instead of computing the paragraph embeddings as a weighted sum of sentence embeddings, we directly obtain the paragraph embeddings from ETC output for this dataset. Recall that a paragraph $p_i = \{s^i_0, \ldots, s^i_{|p_i|}\}$ contains a sequence of sentences s^i_j. We prepend a dummy sentence s_{null} to the beginning of the paragraph, and again, we modify the global-to-local attention mask to allow the global token of the dummy sentence to attend to all tokens in the paragraph p_i. Let $\mathbf{p}_i \in \mathbb{R}^d$ be the embedding of paragraph p_i. The embeddings for a paragraph and its contained sentences are:

$$\mathbf{p}_i, \mathbf{s}^i_0, \ldots, \mathbf{s}^i_{|p_i|} = \text{ETC}(\{s_{\text{null}}, s^i_0, \ldots, s^i_{|p_i|}\})$$

Distant Supervision The iterative attention process is distantly supervised with supervision at intermediate steps. At each step, the model is trained to attend to both the correct paragraph and the correct sentences if they exists. Since the embedding table $\mathbf{C}_d$ consists of both paragraph and sentence embeddings, we only need to compute the attention scores once at each step, but consider both the correct paragraph and the correct sentence as positive. The positive paragraph is one of the paragraphs from the crawled web page with the highest BLEU score.[6] We notice that some web pages at the provided URLs have been changed significantly, so the snippets provided in the datasets may not exist any more, hence we discard the associated data if the highest BLEU scores of the paragraphs is less than 0.7. We follow the heuristics used by baseline models (Gao et al., 2020) to get positive sentence candidates by finding the sentence with the minimum edit distance.

A.4 Additional Results

We report the performance of eventually selecting the correct evidences in Table 5, 6, and 7.

Comments on HotpotQA-Long We also observe that ablated experiment on evidence selection (w/o query update) is only 7.8 points lower than the full model. To understand the underlying reason, we train the model to perform a one-step attention only for supporting facts of the second hop (for bridge questions). The accuracy is 71.7, only 3.6 points lower than the accuracy of the full multi-hop process. This is likely due to the high surface form overlap between the questions and their context.

[6] We drop the brevity penalty term in BLEU score.

	HybridQA	QASPER (Extractive)
DOCHOPPER	**56.5**	**39.1**
(w/o sparse)	53.3	(39.1)
(w/o query update)	51.8	37.2
(sentence-only)	46.4	36.1
(single-hop)	34.2	36.8

Table 5: Hits@1 accuracy on selecting sentences that actually contains the answer (on dev set).

	HotpotQA-Long
IRRR	56.8
DOCHOPPER	64.7
(w/o query update)	56.5
(sentence-only)	61.8
(single-hop)	37.4

Table 6: Accuracy of correctly predicting supporting facts for both hops on HotpotQA-Long (without reranking).

	ShARC-Long
DOCHOPPER	82.2
(w/o query update)	81.8
(sentence-only)	63.0
(single-hop)	72.4

Table 7: Accuracy of selecting all required evidences on ShARC-Long.