Explore, Propose, and Assemble: An Interpretable Model for Multi-Hop Reading Comprehension

Yichen Jiang* **Nitish Joshi*** **Yen-Chun Chen** **Mohit Bansal**
UNC Chapel Hill
{yichenj, nitish, yenchun, mbansal}@cs.unc.edu

Abstract

Multi-hop reading comprehension requires the model to explore and connect relevant information from multiple sentences/documents in order to answer the question about the context. To achieve this, we propose an interpretable 3-module system called Explore-Propose-Assemble reader (EPAr). First, the Document Explorer iteratively selects relevant documents and represents divergent reasoning chains in a tree structure so as to allow assimilating information from all chains. The Answer Proposer then proposes an answer from every root-to-leaf path in the reasoning tree. Finally, the Evidence Assembler extracts a key sentence containing the proposed answer from every path and combines them to predict the final answer. Intuitively, EPAr approximates the coarse-to-fine-grained comprehension behavior of human readers when facing multiple long documents. We jointly optimize our 3 modules by minimizing the sum of losses from each stage conditioned on the previous stage's output. On two multi-hop reading comprehension datasets WikiHop and MedHop, our EPAr model achieves significant improvements over the baseline and competitive results compared to the state-of-the-art model. We also present multiple reasoning-chain-recovery tests and ablation studies to demonstrate our system's ability to perform interpretable and accurate reasoning.[1]

1 Introduction

The task of machine reading comprehension and question answering (MRC-QA) requires the model to answer a natural language question by finding relevant information and knowledge in a given natural language context. Most MRC datasets require single-hop reasoning only, which means that the evidence necessary to answer the question is concentrated in a single sentence or located closely in a single paragraph. Such datasets emphasize the role of locating, matching, and aligning information between the question and the context. However, some recent multi-document, multi-hop reading comprehension datasets, such as WikiHop and MedHop (Welbl et al., 2017), have been proposed to further assess MRC systems' ability to perform multi-hop reasoning, where the required evidence is scattered in a set of supporting documents.

These multi-hop tasks are much more challenging than previous single-hop MRC tasks (Rajpurkar et al., 2016, 2018; Hermann et al., 2015; Nguyen et al., 2016; Yang et al., 2015) for three primary reasons. First, the given context contains a large number of documents (e.g., 14 on average, 64 maximum for WikiHop). Most existing QA models cannot scale to the context of such length, and it is challenging to retrieve a reasoning chain of documents with complete information required to connect the question to the answer in a logical way. Second, given a reasoning chain of documents, it is still necessary for the model to consider evidence loosely distributed in all these documents in order to predict the final answer. Third, there could be more than one logical way to connect the scattered evidence (i.e., more than one possible reasoning chain) and hence this requires models to assemble and weigh information collected from every reasoning chain before making a unified prediction.

To overcome the three difficulties elaborated above, we develop our interpretable 3-module system based on examining how a human reader would approach a question, as shown in Fig. 1a and Fig. 1b. For the 1st example, instead of reading the entire set of supporting documents sequen-

*equal contribution; part of this work was done during the second author's internship at UNC (from IIT Bombay).

[1]Our code is publicly available at: https://github.com/jiangycTarheel/EPAr

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 2714–2725
Florence, Italy, July 28 - August 2, 2019.

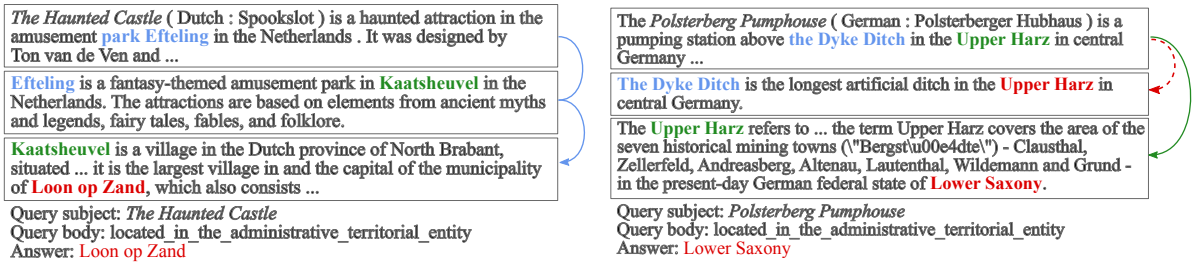

Figure 1: Two examples from the QAngaroo WikiHop dataset where it is necessary to combine information spread across multiple documents to infer the correct answer. **(a)**: The hidden reasoning chain of 3 out of a total of 37 documents for a single query. **(b)**: Two possible reasoning chains that lead to different answers: "Upper Harz" and "Lower Saxony", while the latter (green solid arrow) fits better with query body "administrative territorial entity".

tially, she would start from the document that is directly related to the query subject (e.g., "The Haunted Castle"). She could then read the second and third document by following the connecting entities "park Efteling" and "Kaatsheuvel", and uncover the answer "Loon op Zand" by comparing phrases in the final document to the query. In this way, the reader accumulates knowledge about the query subject by exploring inter-connected documents, and eventually uncovers the entire reasoning chain that leads to the answer. Drawing inspiration from this coarse (document-level) plus fine-grained (word-level) comprehension behavior, we first construct a T-hop Document Explorer model, a hierarchical memory network, which at each recurrent hop, selects one document to read, updates the memory cell, and iteratively selects the next related document, overall constructing a reasoning chain of the most relevant documents. We next introduce an Answer Proposer that performs query-context reasoning at the word-level on the retrieved chain and predicts an answer. Specifically, it encodes the leaf document of the reasoning chain while attending to its ancestral documents, and outputs ancestor-aware word representations for this leaf document, which are compared to the query to propose a candidate answer.

However, these two components above cannot handle questions that allow multiple possible reasoning chains that lead to different answers, as shown in Fig. 1b. After the Document Explorer selects the 1st document, it finds that both the 2nd and 3rd documents are connected to the 1st document via entities "the Dyke Ditch" and "Upper Harz" respectively. This is a situation where a single reasoning chain diverges into multiple paths, and it is impossible to tell which path will lead to the correct answer before finishing exploring all possible reasoning chains/paths. Hence, to be able to weigh and combine information from multiple reasoning branches, the Document Explorer is rolled out multiple times to represent all the divergent reasoning chains in a 'reasoning tree' structure, so as to allow our third component, the Evidence Assembler, to assimilate important evidence identified in every reasoning chain of the tree to make one final, unified prediction. To do so, the Assembler selects key sentences from each root-to-leaf document path in the 'reasoning tree' and forms a new condensed, salient context which is then bidirectionally-matched with the query representation to output the final prediction. Via this procedure, evidence that was originally scattered widely across several documents is now collected concentratedly, hence transforming the task to a scenario where previous standard phrase-matching style QA models (Seo et al., 2017; Xiong et al., 2017; Dhingra et al., 2017) can be effective.

Overall, our 3-module, multi-hop, reasoning-tree based EPAr (Explore-Propose-Assemble reader) closely mimics the coarse-to-fine-grained reading and reasoning behavior of human readers. We jointly optimize this 3-module system by having the following component working on the outputs from the previous component and minimizing the sum of the losses from all 3 modules. The Answer Proposer and Evidence Assembler are trained with maximum likelihood using ground-truth answers as labels, while the Document Explorer is weakly supervised by heuristic reasoning chains constructed via TF-IDF and documents with the ground-truth answer.

On WikiHop, our system achieves the highest-reported dev set result of 67.2%, outperforming all published models[2] on this task, and 69.1%

[2] At the time of submission: March 3rd, 2019.

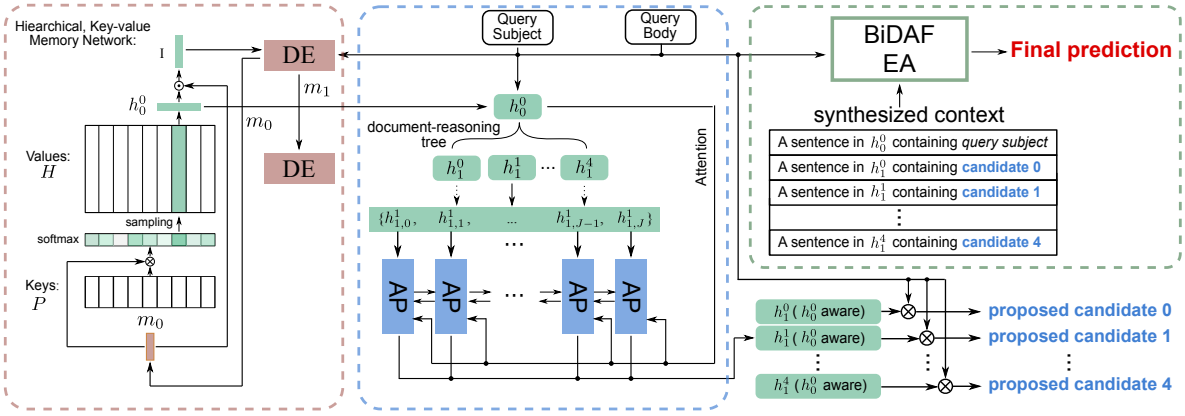

Figure 2: The full architecture of our 3-module system EPAr, with the Document Explorer (DE, left), Answer Proposer (AP, middle), and Evidence Assembler (EA, right).

accuracy on the hidden test set, which is competitive with the current leaderboard state-of-the-art. On MedHop, our system outperforms all previous models, achieving the new state-of-the-art test leaderboard accuracy. It also obtains statistically significant ($p < 0.01$) improvement over our strong baseline on the two datasets. Further, we show that our Document Explorer combined with 2-hop TF-IDF retrieval is substantially better than two TF-IDF-based retrieval baselines in multiple reasoning-chain recovery tests including on human-annotated golden reasoning chains. Next, we conduct ablations to prove the effectiveness of the Answer Proposer and Evidence Assembler in comparison with several baseline counterparts, and illustrate output examples of our 3-module system's reasoning tree.

2 Model

In this section, we describe our 3-module system that constructs the 'reasoning tree' of documents and predicts the answer for the query. Formally, given a query q and a corresponding set of supporting documents $D = \{d_i\}_{i=1}^{N}$, our system tries to find a reasoning chain of documents $d'_1, \ldots, d'_T, d'_i \in D$.[3] The information from these selected documents is then combined to predict the answer among the given answer candidates. In the WikiHop and MedHop datasets, a query consists of a subject q_{sub} (e.g., "The Haunted Castle" in Fig. 1a) and a body q_{bod} (e.g., "located in the administrative territorial entity"). There is one single correct answer a (e.g., "Loon op Zand") in the set of candidate answers $A = \{c_l\}_{l=1}^{L}$ such that the relation q_{bod} holds true between q_{sub} and a.

[3] In WikiHop dataset, $T \leq 3$.

2.1 Retrieval and Encoding

In this section, we describe the pre-processing document retrieval and encoding steps before introducing our three modules of EPAr. We adopt a 2-hop document retrieval procedure to reduce the number of supporting documents that are fed to our system. We first select one document with the shortest TF-IDF distance to the query. We then rank the remaining documents according to their TF-IDF distances to the first selected document and add the top $N'-1$ documents to form the context with a total of N' documents for this query. Adding this preprocessing step is not only helpful in reducing GPU memory consumption but also helps bootstrap the training by reducing the search space of the Document Explorer (Sec. 2.2).

We then use a Highway Network (Srivastava et al., 2015) of dimension d, which merges the character embedding and GloVe word embedding (Pennington et al., 2014), to get the word representations for the supporting documents and query[4]. This gives three matrices: $\mathbf{X} \in \mathbb{R}^{N' \times K \times d}$, $\mathbf{Q_{sub}} \in \mathbb{R}^{J_s \times d}$ and $\mathbf{Q_{bod}} \in \mathbb{R}^{J_b \times d}$, K, J_s, J_b are the lengths of supporting documents, query body, and query subject respectively. We then apply a bi-directional LSTM-RNN (Hochreiter and Schmidhuber, 1997) of v hidden units to get the contextual word representations for the documents $\mathbf{H} = \{h_1, \cdots, h_{N'}\}$ s.t. $h_i \in \mathbb{R}^{K \times 2v}$ and the query $\mathbf{U_{sub}} \in \mathbb{R}^{J_s \times 2v}$, $\mathbf{U_{bod}} \in \mathbb{R}^{J_b \times 2v}$. Other than the word-level encoding, we also collect compact representations of all the supporting docu-

[4] Unlike previous works (Welbl et al., 2017; Dhingra et al., 2018; De Cao et al., 2018; Song et al., 2018a) that concatenate supporting documents together to form a large context, we instead maintain the document-level hierarchy and encode each document separately.

ments, denoted as $\mathbf{P} = \{p_1, \cdots, p_{N'}\}$, by applying the self-attention mechanism in Zhong et al. (2019) (see details in appendix). We obtain embeddings for each candidate $c_i \in \{c_1, c_2, .., c_L\}$ using the average-over-word embeddings of the first mention[5] of the candidate in $\mathbf{H}$.

2.2 Document Explorer

Our Document Explorer (DE, shown in the left part of Fig. 2) is a hierarchical memory network (Chandar et al., 2016). It utilizes the reduced document representations $\mathbf{P} = \{p_1, p_2, \cdots, p_{N'}\}$ and their corresponding word-level representations $\mathbf{H} = \{h_1, h_2, \cdots, h_{N'}\}$ as the key-value knowledge base and maintains a memory m using a Gated Recurrent Unit (GRU) (Cho et al., 2014). At every step, the DE selects a document which is related to the current memory state and updates the internal memory. This iterative procedure thus constructs a reasoning chain of documents.

Read Unit At each hop t, the model computes a *document-selection distribution* P over every document based on the bilinear-similarity between the memory state m and document representations $\mathbf{P}$ using the following equations[6]:

$$x_n = p_n^T \mathbf{W_r} m^t \quad \chi = \text{softmax}(x) \quad P(d_i) = \chi_i$$

The read unit looks at all document (representation) $\mathbf{P}$ and selects (samples) a document $d_i \sim P$. The write operation updates the internal state (memory) using this sampled document.

Write Unit After the model selects $d_i \in D$, the model then computes a distribution over every word in document d_i based on the similarity between the memory state $\mathbf{m}$ and its word representations $h_i \in \mathbf{H}$. This distribution is then used to compute the weighted average of all word representations in document d_i. We then feed this weighted average $\tilde{h}$ as the input to the GRU cell and update its memory state $\mathbf{m}$ (subscript i is omitted for simplicity):

$$\begin{aligned} w_k &= h_k^T \mathbf{W_w} m & \tilde{h} &= \sum_{k=1}^{K} h_k \omega_k \\ \omega &= \text{softmax}(w) & m^{t+1} &= \mathbf{GRU}(\tilde{h}, m^t) \end{aligned} \quad (1)$$

Combining the 'read' and 'write' operations described above, we define a recurrent function: $(\hat{h}_{t+1},\ m^{t+1}) = f_{DE}(m^t)$ such that $\hat{h}_{t+1} \in \mathbf{H}$ and $\hat{h}_t \neq \hat{h}_{t+1}$. Therefore, unrolling the Document Explorer for T hops results in a sequence of *non-repeating* documents $\hat{\mathbf{H}} = \{\hat{h}_1, \cdots, \hat{h}_T\}$ such that each document $\hat{h}_i$ is selected iteratively based on the current memory state building up one reasoning chain of documents. In practice, we roll out DE multiple times to obtain a document-search 'reasoning tree', where each root-to-leaf path corresponds to a query-to-answer reasoning chain.

2.3 Answer Proposer

The Answer Proposer (AP, shown in the middle part of Fig. 2) takes as input a single chain of documents $\{\hat{h}_1, \cdots, \hat{h}_T\}$ from one of the chains in the 'reasoning tree' created by the DE, and tries to predict a candidate answer from the last document $\hat{h}_T$ in that reasoning chain. Specifically, we adopt an LSTM-RNN with an attention mechanism (Bahdanau et al., 2015) to encode the $\hat{h}_T$ to ancestor-aware representations y by attending to $[\hat{h}_{1,\ldots,T-1}]$. The model then computes a distribution over words $\hat{h}_T^i \in \hat{h}_T$ based on the similarity between y and the query representation. This distribution is then used to compute the weighted average of word representations $\{h_T^1, h_T^2, \cdots, h_T^K\}$. Finally, AP proposes an answer among all candidates $\{c_1, \cdots, c_L\}$ that has the largest similarity score with this weighted average $\tilde{h}_T$.

$$\begin{aligned} e_i^k &= \mathbf{v}^T \tanh(\mathbf{W_h} \hat{h}_{cct}^i + \mathbf{W_s} s^k + \mathbf{b}) \\ a^k &= \text{softmax}(e^k); \quad c^k = \sum_i a_i^k h_{cct}^i \\ y^k &= \mathbf{LSTM}(\hat{h}_T^{k-1}, s^{k-1}, c^{k-1}) \\ w^k &= \boldsymbol{\alpha}(y^k, u_s) + \boldsymbol{\alpha}(y^k, u_b); \quad \epsilon = \text{softmax}(w) \\ a &= \sum_{k=1}^{K} \hat{h}_T^k \epsilon_k; \quad Score_l = \boldsymbol{\beta}(c_l, a) \end{aligned} \quad (2)$$

where $\hat{h}_{cct} = [\hat{h}_{1,\ldots,T-1}]$ is the concatenation of documents in the word dimension; u_s and u_b are the final states of $\mathbf{U_{sub}}$ and $\mathbf{U_{bod}}$ respectively, and s^k is the LSTM's hidden states at the kth step. The Answer Proposer proposes the candidate with the highest score among $\{c_1, \cdots, c_L\}$. All computations in Eqn. 2 that involve trainable parameters are marked in bold.[7] This procedure produces ancestor-aware word representations that encode the interactions between the leaf document and ancestral document, and hence models the multi-hop, cross-document reasoning behavior.

[5] We tried different approaches to make use of all mentions of every candidate, but observe no gain in final performance.

[6] We initialize the memory state with the last state of the query subject $\mathbf{U_{sub}}$ to make first selected document directly conditioned on the query subject.

[7] See appendix for the definition of the similarity functions $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$.

2.4 Evidence Assembler

As shown in Fig. 1b, it is possible that a reasoning path could diverge into multiple branches, where each branch represents a unique, logical way of retrieving inter-connected documents. Intuitively, it is very difficult for the model to predict which path to take without looking ahead. To solve this, our system first explores multiple reasoning chains by rolling out the Document Explorer multiple times to construct a 'reasoning tree' of documents, and then aggregates information from multiple reasoning chains using a Evidence Assembler (EA, shown in the right part of Fig. 2), to predict the final answer. For each reasoning chain, the Assembler first selects one sentence that contains the candidate answer proposed by the Answer Proposer and concatenates all these sentences into a new document h'. This constructs a highly informative and condensed context, at which point previous phrase-matching style QA models can work effectively. Our EA uses a bidirectional attention flow model (Seo et al., 2017) to get a distribution over every word in h' and compute the weighted average of word representations $\{h'^1, \cdots, h'^K\}$ as $\tilde{h}'$. Finally, the EA selects the candidate answer of the highest similarity score w.r.t. $\tilde{h}'$.

2.5 Joint Optimization

Finally, we jointly optimize the entire model using the cross-entropy losses from our Document Explorer, Answer Proposer, and Evidence Assembler. Since the Document Explorer samples documents from a distribution, we use weak supervision at the first and the final hops to account for the otherwise non-differentiabilty in the case of end-to-end training. Specifically, we use the document having the shortest TF-IDF distance w.r.t. the query subject to supervise the first hop and the documents which contain at least one mention of the answer to supervise the last hop. This allows the Document Explorer to learn the chain of documents leading to the document containing the answer from the document most relevant to the query subject. Since there can be multiple documents containing the answer, we randomly sample a document as the label at the last hop. For the Answer Proposer and Evidence Assembler, we use cross-entropy loss from the answer selection process.

3 Experiments and Results

3.1 Datasets and Metrics

We evaluate our 3-module system on the WikiHop and the smaller MedHop multi-hop datasets from QAngaroo (Welbl et al., 2017). For the WikiHop dev set, each instance is also annotated as "follows" or "not follows", i.e., whether the answer can be inferred from the given set of supporting documents, and "single" or "multiple", indicating whether the complete reasoning chain comprises of single or multiple documents. This allows us to evaluate our system on less noisy data and to investigate its strength in queries requiring different levels of multi-hop reasoning. Please see appendix for dataset and metric details.

3.2 Implementation Details

For WikiHop experiments, we use 300-d GloVe word embeddings (Pennington et al., 2014) for our main full-size 'EPAr' model and 100-d GloVE word embeddings for our smaller 'EPAr' model which we use throughout the Analysis section for time and memory feasibility. We also use the last hidden state of the encoding LSTM-RNN to get the compact representation for all supporting documents in case of smaller model, in contrast to self-attention (Sec. B in Appendix) as in the full-size 'EPAr' model. The encoding LSTM-RNN (Hochreiter and Schmidhuber, 1997) has 100-d hidden size for our 'EPAr' model whereas the smaller version has 20-d hidden size. The embedded GRU (Cho et al., 2014) and the LSTM in our Evidence Assembler have the hidden dimension of 80. In practice, we only apply TF-IDF based retrieval procedure to our Document Explorer and Answer Proposer during inference, and during training time we use the full set of supporting documents as the input. This is because we observed that the Document Explorer overfits faster in the reduced document-search space. For the Evidence Assembler, we employ both the TF-IDF retrieval and Document Explorer to get the 'reasoning tree' of documents, at both training and testing time. We refer to the Sec. E in the appendix for the implementation details of our MedHop models.

3.3 Results

We first evaluate our system on the WikiHop dataset. For a fair comparison to recent works (De Cao et al., 2018; Song et al., 2018a; Raison et al., 2018), we report our "EPAr" with

	Dev	Test
BiDAF (Welbl et al., 2017)⋆	-	42.9
Coref-GRU (Dhingra et al., 2018)	56.0	59.3
WEAVER (Raison et al., 2018)	64.1	65.3
MHQA-GRN (Song et al., 2018a)	62.8	65.4
Entity-GCN (De Cao et al., 2018)	64.8	67.6
BAG (Cao et al., 2019)	66.5	69.0
CFC (Zhong et al., 2019)	66.4	70.6
EPAr (Ours)	**67.2**	69.1

Table 1: Dev set and Test set accuracy on WIKIHOP dataset. The model marked with ⋆ does not use candidates and directly predict the answer span. EPAr is our system with TF-IDF retrieval, Document Explorer, Answer Proposer and Evidence Assembler.

	follow + multiple	follow + single	full
BiDAF Baseline	62.8	63.1	58.4
DE+AP+EA⋆	65.2	66.9	61.1
AP+EA	68.7	67.0	62.8
DE+AP+EA	69.4	70.6	64.7
DE+AP+EA†	71.8	**73.8**	66.9
DE+AP+EA†+SelfAttn	**73.5**	72.9	**67.2**

Table 2: Ablation accuracy on WIKIHOP dev set. The model marked with ⋆ does not use the TFIDF-based document retrieval procedure. The models marked with † are our full EPAr systems with 300-d word embeddings and 100-d LSTM-RNN hidden size (same as the last row of Table 1), while the 4th row represents the smaller EPAr system.

300-d embeddings and 100-d hidden size of the encoding LSTM-RNN. As shown in Table 1, EPAr achieves 67.2% accuracy on the dev set, outperforming all published models, and achieves 69.1% accuracy on the hidden test set, which is competitive with the current state-of-the-art result.[8]

Next, in Table 2, we further evaluate our EPAr system (and its smaller-sized and ablated versions) on the "follows + multiple", "follows + single", and the full development set. First, note that on the full development set, our smaller system ("DE+AP+EA") achieves statistically significant (p-value < 0.01)[9] improvements over the BiDAF baseline and is also comparable to De Cao et al. (2018) on the development set (64.7 vs. 64.8).[10]

[8]Note that there also exists a recent anonymous unpublished entry on the leaderboard with 70.9% accuracy, which is concurrent to our work. Also note that our system achieves these strong accuracies even without using pretrained language model representations like ELMo (Peters et al., 2018) or BERT (Devlin et al., 2018), which have been known to give significant improvements in machine comprehension and QA tasks. We leave these gains for future work.

[9]All stat. signif. is based on bootstrapped randomization test with 100K samples (Efron and Tibshirani, 1994).

[10]For time and memory feasibility, we use this smaller strong model with 100-d word embeddings and 20-d LSTM-RNN hidden size (similar to baselines in Welbl et al. (2017)) in all our analysis/ablation results (including Sec. 4).

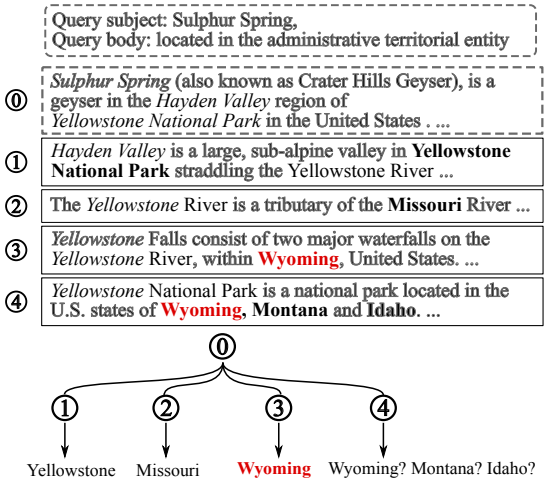

Figure 3: A 'reasoning tree' with 4 leaves that lead to different answers (marked in bold). The ground-truth answer is marked in red additionally.

Moreover, we see that EPAr is able to achieve high accuracy in both the examples that require multi-hop reasoning ("follows + multiple"), and other cases where a single document suffices for correctly answering the question ("follows + single"), suggesting that our system is able to adjust to examples of different reasoning requirements. The evaluation results further demonstrate that our Document Explorer combined with TF-IDF-based retrieval (row 'DE+AP+EA') consistently outperforms TF-IDF alone (row 'AP+EA') or the Document Explorer without TF-IDF (row 'DE+AP+EA⋆' in Table 2), showing that our 2-hop TF-IDF document retrieval procedure is able to broadly identify relevant documents and further aid our Document Explorer by reducing its search space. Finally, comparing the last two rows in Table 2 shows that using self-attention (Zhong et al., 2019) to compute the document representation can further improve the full-sized system. We show an example of the 'reasoning tree' constructed by the Document Explorer and the correct answer predicted by the Evidence Assembler in Fig. 3.

We report our system's accuracy on the MedHop dataset in Table 3. Our best system achieves 60.3 on the hidden test set[11], outperforming all current models on the leaderboard. However, as reported by Welbl et al. (2017), the original MedHop dataset suffers from a candidate frequency imbalance issue that can be exploited by certain

[11]The masked MedHop test set results use the smaller size model, because this performed better on the masked dev set.

	Test (Masked)	Test
FastQA* (Weissenborn et al., 2017)	23.1	31.3
BiDAF* (Seo et al., 2017)	33.7	47.8
CoAttention	-	58.1
Most Frequent Candidate*	10.4	58.4
EPAr (Ours)	**41.6**	**60.3**

Table 3: Test set accuracy on MEDHOP dataset. The results marked with * are reported in (Welbl et al., 2017).

	R@1	R@2	R@3	R@4	R@5
Random	11.2	17.3	27.6	40.8	50.0
1-hop TFIDF	32.7	48.0	56.1	63.3	70.4
2-hop TFIDF	42.9	56.1	70.4	78.6	82.7
DE	38.8	50.0	65.3	73.5	83.7
TFIDF+DE	**44.9**	**64.3**	**77.6**	**82.7**	**90.8**

Table 4: Recall-k score is the % of examples where one of the human-annotated reasoning chains is recovered in the top-k root-to-leaf paths in the 'reasoning tree'. 'TFIDF+DE' is the combination of the 2-hop TF-IDF retrieval procedure and our Document Explorer.

heuristics like the 'Most Frequent Candidate' in Table 3. To eliminate this bias and to test our system's ability to conduct multi-hop reasoning using the context, we additionally evaluate our system on the masked version of MedHop, where every candidate expression is replaced randomly using 100 unique placeholder tokens so that models can only rely on the context to comprehend every candidate. Our model achieves 41.6% accuracy in this "masked" setting, outperforming all previously published works by a large margin.

4 Analysis

In this section, we present a series of new analyses and comparisons in order to understand the contribution from each of our three modules and demonstrate their advantages over other corresponding baselines and heuristics.

4.1 Reasoning Chain Recovery Tests

We compare our Document Explorer with two TF-IDF-based document selectors for their ability to recover the reasoning chain of documents. The 1-hop TF-IDF selector selects the top $k + 1$ documents with the highest TF-IDF score w.r.t. the query subject. The 2-hop TF-IDF selector, as in Sec. 2.1, first selects the top-1 TF-IDF document w.r.t. the query subject and then selects the top k remaining documents based on the TF-IDF score with respect to the first selected document. Finally, we also compare to our final combination of 2-hop TF-IDF and Document Explorer.

	R@1	R@2	R@3	R@4	R@5
Random	39.9	51.4	60.2	67.8	73.5
1-hop TFIDF	38.4	48.5	58.6	67.4	73.7
2-hop TFIDF	38.4	58.7	70.2	77.2	81.6
DE	**52.5**	**70.2**	**80.3**	**85.8**	**89.0**
TFIDF+DE	52.2	69.0	77.8	82.2	85.2

Table 5: Recall-k score is the percentage of examples where the ground-truth answer is present in the top-k root-to-leaf path in the 'reasoning tree'. 'TFIDF+DE' is the combination of the 2-hop TFIDF retrieval procedure and our Document Explorer.

Human Evaluation: We collect human-annotated reasoning chains for 100 documents from the "follows + multiple" dev set, and compare these to the 'reasoning tree' constructed by our Document Explorer to assess its ability to discover the hidden reasoning chain from the entire pool of supporting documents. For each example, human annotators (external, English-speaking) select two of the smallest set of documents, from which they can reason to find the correct answer from the question. As shown in Table 4, our Document Explorer combined with 2-hop TF-IDF (row 'TFIDF+DE') obtains higher golden-chain recall scores compared to the two TFIDF-based document retrieval heuristics (row '1-hop TFIDF' and '2-hop TFIDF') alone or the Document Explorer without TF-IDF (row 'DE').

Answer Span Test: We also test our Document Explorer's ability to find the document with mentions of the ground-truth answer. Logically, the fact that the answer appears in one of the documents in the 'reasoning tree' signals higher probability that our modules at the following stages could predict the correct answer. As shown in Table 5, our Document Explorer receives significantly higher answer-span recall scores compared to the two TF-IDF-based document selectors.[12]

4.2 Answer Proposer Comparisons

We compare our Answer Proposer with two rule-based sentence extraction heuristics for the ability to extract salient information from every reasoning chain. For most documents in the WikiHop dataset, the first sentence is comprised of the most salient information from that document. Hence,

[12] In this test, the Document Explorer alone outperforms its combination with the 2-hop TF-IDF retrieval. In practice, our system employs both procedures due to the advantage shown in both empirical results (Table 2) and analysis (Table 4).

	full	follows + multiple	follows + single
Full-doc	63.1	68.4	69.0
Lead-1	63.6	68.7	70.2
AP w.o. attn	63.3	68.3	69.6
AP	**64.7**	**69.4**	**70.6**

Table 6: Answer Proposer comparison study. "Follows + multiple" and "follows + single" are the subsets of dev set as described in Sec. 3.1.

	full	follows + multiple	follows + single
Single-chain	59.9	64.3	63.8
Avg-vote	54.6	56.3	55.6
Max-vote	51.5	53.9	53.3
w. Reranker	60.6	65.1	65.5
w. Assembler	**64.7**	**69.4**	**70.6**

Table 7: Evidence Assembler comparison study: Reranker (described in the appendix) rescores the documents selected by the Document Explorer.

we construct one baseline that concatenates the first sentence from each selected document as the input to the Evidence Assembler. We also show results of combining all the full documents as the synthesized context instead of selecting one sentence from every document. We further present a lighter neural-model baseline that directly proposes the answer from the leaf document without first creating its ancestor-aware representation. As shown in Table 6, the system using sentences selected by our Answer Proposer outperforms both rule-based heuristics (row 1 and 2) and the simple neural baseline (row 3).

4.3 Assembler Ablations

In order to justify our choice of building an Assembler, we build a 2-module system without the Evidence-Assembler stage by applying the Answer Proposer to only the top-1 reasoning chain in the tree. We also present two voting heuristics that selects the final answer by taking the average/maximum prediction probability from the Answer Proposer on all document chains. Furthermore, we compare our Evidence Assembler with an alternative model that, instead of assembling information from all reasoning chains, reranks all chains and their proposed answers to select the top-1 answer prediction. As shown in Table 7, the full system with the Assembler achieves significant improvements over the 2-module system. This demonstrates the importance of the Assembler in enabling information aggregation over multiple reasoning chains. The results further show that our Assembler is better than the reranking alternative.

4.4 Multi-hop Reasoning Example

We visualize the 3-stage reasoning procedure of our EPAr system in Fig. 4. As shown in the left of Fig. 4, the Document Explorer first locates the root document ("The Polsterberg Pumphouse ...") based on the query subject. It then finds three more documents that are related to the root document, constructing three document chains. The Answer Proposer proposes a candidate answer from each of the three chains selected by the Document Explorer. Finally, the Evidence Assembler selects key sentences from all documents in the constructed document chains and makes the final prediction ("Lower Saxony").

5 Related Works

The last few years have witnessed significant progress on text-based machine reading comprehension and question answering (MRC-QA) including cloze-style blank-filling tasks (Hermann et al., 2015), open-domain QA (Yang et al., 2015), answer span prediction (Rajpurkar et al., 2016, 2018), and generative QA (Nguyen et al., 2016). However, all of the above datasets are confined to a single-document context per question setup. Joshi et al. (2017) extended the task to the multi-document regime, with some examples requiring cross-sentence inference. Earlier attempts in multi-hop MRC focused on reasoning about the relations in a knowledge base (Jain, 2016; Zhou et al., 2018; Lin et al., 2018) or tables (Yin et al., 2015). QAngaroo WikiHop and MedHop (Welbl et al., 2017), on the other hand, are created as *natural language* MRC tasks. They are designed in a way such that the evidence required to answer a query could be spread across multiple documents. Thus, finding some evidence requires building a reasoning chain from the query with intermediate inference steps, which poses extra difficulty for MRC-QA systems. HotpotQA (Yang et al., 2018) is another recent multi-hop dataset which focuses on four different reasoning paradigms.

The emergence of large-scale MRC datasets has led to innovative neural models such as co-attention (Xiong et al., 2017), bi-directional attention flow (Seo et al., 2017), and gated attention (Dhingra et al., 2017), all of which are metic-

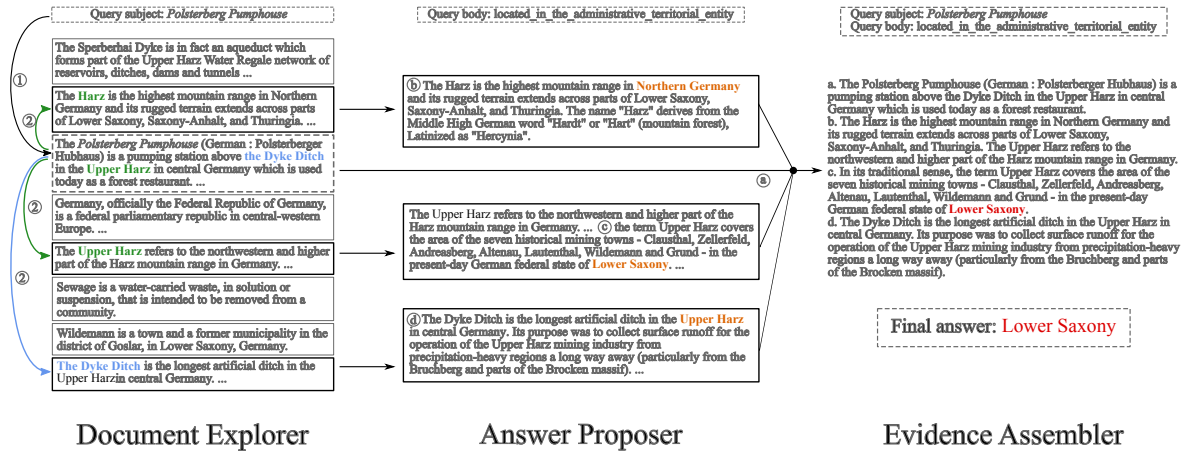

Figure 4: An example of our 3-stage EPAr system exploring relevant documents, proposing candidate answers, and then assembling extracted evidence to make the final prediction.

ulously designed to solve single-document MRC tasks. Clark and Gardner (2018) and Chen et al. (2017) used a simple TF-IDF based document-selection procedure to find the context that is most relevant to the query for multi-document QA. However, this 1-hop, similarity-based selection process would fail on multi-hop reading-comprehension datasets like WikiHop because the query subject and the answer could appear in different documents. On the other hand, our Document Explorer can discover the document with the answer "Loon op Zand" (in Fig. 1a) by iteratively selecting relevant documents and encoding the hinge words "Efteling" and "Kaatsheuvel" in its memory.

Recently, Dhingra et al. (2018) leveraged coreference annotations from an external system to connect the entities. Song et al. (2018a) and De Cao et al. (2018) utilized Graph Convolutional Networks (Kipf and Welling, 2017) and Graph Recurrent Networks (Song et al., 2018b; Zhang et al., 2018) to model the relations between entities. Recently, Cao et al. (2019) extended the Graph Convolutional Network in De Cao et al. (2018) by introducing bi-directional attention between the entity graph and query. By connecting the entities, these models learn the inference paths for multi-hop reasoning. Our work differs in that our system learns the relation implicitly without the need of any human-annotated relation. Recently, Zhong et al. (2019) used hierarchies of co-attention and self-attention to combine evidence from multiple scattered documents. Our novel 3-module architecture is inspired by previous 2-module selection architectures for MRC (Choi et al., 2017). Similarly, Wang et al. (2018) first selected relevant content by ranking documents and then extracted the answer span. Min et al. (2018) selected relevant sentences from long documents in a single-document setup and achieved faster speed and robustness against adversarial corruption. However, none of these models are built for multi-hop MRC where our EPAr system shows great effectiveness.

6 Conclusion

We presented an interpretable 3-module, multi-hop, reading-comprehension system 'EPAr' which constructs a 'reasoning tree', proposes an answer candidate for every root-to-leaf chain, and merges key information from all reasoning chains to make the final prediction. On WikiHop, our system outperforms all published models on the dev set, and achieves results competitive with the current state-of-the-art on the test set. On MedHop, our system outperforms all previously published models on the leaderboard test set. We also presented multiple reasoning-chain recovery tests for the explainability of our system's reasoning capabilities.

7 Acknowledgement

We would like to thank Johannes Welbl for helping test our system on WikiHop and MedHop. We thank the reviewers for their helpful comments. This work was supported by DARPA (YFA17-D17AP00022), Google Faculty Research Award, Bloomberg Data Science Research Grant, Salesforce Deep Learning Research Grant, Nvidia GPU awards, Amazon AWS, and Google Cloud Credits. The views contained in this article are those of the authors and not of the funding agency.

References

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Third International Conference on Learning Representations*.

Yu Cao, Meng Fang, and Dacheng Tao. 2019. BAG: bi-directional attention entity graph convolutional network for multi-hop reasoning question answering. In *NAACL-HLT*.

Sarath Chandar, Sungjin Ahn, Hugo Larochelle, Pascal Vincent, Gerald Tesauro, and Yoshua Bengio. 2016. Hierarchical memory networks. *arXiv preprint arXiv:1605.07427*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *ACL*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*.

Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. 2017. Coarse-to-fine question answering for long documents. In *ACL*.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Bhuwan Dhingra, Qiao Jin, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2018. Neural models for reasoning over multiple mentions using coreference. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2017. Gated-attention readers for text comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1832–1846, Vancouver, Canada. Association for Computational Linguistics.

Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Sarthak Jain. 2016. Question answering over knowledge base using factual memory networks. In *Proceedings of the NAACL Student Research Workshop*. Association for Computational Linguistics.

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*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. In *EMNLP*.

Sewon Min, Victor Zhong, Richard Socher, and Caiming Xiong. 2018. Efficient and robust question answering from minimal context over documents. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1725–1735. Association for Computational Linguistics.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke S. Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*.

Martin Raison, Pierre-Emmanuel Mazaré, Rajarshi Das, and Antoine Bordes. 2018. Weaver: Deep co-encoding of questions and documents for machine reading. *arXiv preprint arXiv:1804.10490*.

P. Rajpurkar, R. Jia, and P. Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics (ACL)*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations (ICLR)*.

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

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018b. A graph-to-sequence model for amr-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1616–1626. Association for Computational Linguistics.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. In *International Conference on Machine Learning (ICML)*.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2018. R3: Reinforced ranker-reader for open-domain question answering. In *AAAI*.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Making neural qa as simple as possible but not simpler. In *CoNLL*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2017. Constructing datasets for multi-hop reading comprehension across documents. In *TACL*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. In *ICLR*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018.

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)*.

Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2015. Neural enquirer: Learning to query tables. *arXiv preprint*.

Yue Zhang, Qi Liu, and Linfeng Song. 2018. Sentence-state lstm for text representation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Victor Zhong, Caiming Xiong, Nitish Shirish Keskar, and Richard Socher. 2019. Coarse-grain fine-grain coattention network for multi-evidence question answering. In *ICLR*.

Mantong Zhou, Minlie Huang, and Xiaoyan Zhu. 2018. An interpretable reasoning network for multi-relation question answering. In *Proceedings of the 27th International Conference on Computational Linguistics*.

Appendix

A Reranker

We explore an alternative to Evidence Assembler (EA), where instead of selecting key sentences from every root-to-leaf path in the reasoning tree, we use a reranker to rescore the selected documents. Specifically, given a document reasoning-tree of t_w reasoning chains, we use bidirectional attention (Seo et al., 2017) between the last documents in each chain and all the documents from the previous hops in that chain to obtain $\{\hat{h}_1, \cdots, \hat{h}_{t_w}\}$ which are the refined representations of the leaf documents. We then obtain a fixed length document representation as the weighted average of word representations for each of the t_w documents using similarity with query subject and query body as the weights using function $\boldsymbol{\alpha}$. We obtain the scores for each of the documents by computing similarity with the answer which that reasoning chain proposes using $\boldsymbol{\beta}$. (See Sec. C below for details of the similarity functions $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$.)

B Self-Attention

We use self-attention from Zhong et al. (2019) to get the compact representation for all supporting documents. Given contextual word representations for the supporting documents $\mathbf{H} = \{h_1, h_2, \cdots, h_{N'}\}$ such that $h_i \in \mathbb{R}^{K \times 2v}$, we define $\text{Selfattn}(h_i) \rightarrow p_i \in \mathbb{R}^{2v}$ as:

$$\begin{aligned} a_{ik} &= \tanh(W_2 \tanh(W_1 h_i^k + b1) + b2) \\ \hat{a}_i &= \text{softmax}(a_i) \\ p_i &= \sum_{k=1}^{K} \hat{a}_{ik} h_i^k \end{aligned} \tag{3}$$

such that p_i provides the summary of the ith document with a vector representation.

C Similarity Functions

When constructing our 3-module system, we use similarity functions $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. The function $\boldsymbol{\beta}$ is defined as:

$$\boldsymbol{\beta}(h, c) = \mathbf{W}_{\boldsymbol{\beta_1}} \text{relu}(\mathbf{W}_{\boldsymbol{\beta_2}}[h; u; h \circ u] + \mathbf{b}_{\boldsymbol{\beta_2}}) + \mathbf{b}_{\boldsymbol{\beta_1}} \quad (4)$$

where $\text{relu}(x) = \max(0, x)$, and $\circ$ represents element-wise multiplication. And the function $\boldsymbol{\alpha}$ is defined as:

$$\boldsymbol{\alpha}(h, u) = \mathbf{W}_{\boldsymbol{\alpha_2}}{}^{T}((\mathbf{W}_{\boldsymbol{\alpha_1}} h + \mathbf{b}_{\boldsymbol{\alpha_1}}) \circ u) \quad (5)$$

where all trainable weights are marked in bold.

D Datasets and Metrics

We evaluate our 3-module system on QAngaroo (Welbl et al., 2017), which is a set of two multi-hop reading comprehension datasets: WikiHop and MedHop. WikiHop contains 51K instances, including 44K for training, 5K for development and 2.5K for held out testing. MedHop is a smaller dataset based on the domain of molecular biology. It consists of 1.6K instances for training, 342 for development, and 546 for held out testing. Each instance consists of a query (which can be separated as a query subject and a query body), a set of supporting documents and a list of candidate answers. For the WikiHop development set, each instance is also annotated as "follows" or "not follows", which signifies whether the answer can be inferred from the given set of supporting documents, and "multiple" or "single", which tells whether the complete reasoning chain comprises of multiple documents or just a single one. We measure our system's performance on these subsets of the development set that are annotated as "follows and multiple" and "follows and single". This allows us to evaluate our systems on a less noisy version of development set and to investigate their strength in queries requiring different levels of multi-hop reasoning behavior.

E Implementation Details

For Medhop, considering the small size of the dataset, we use 20-d hidden size of the encoding LSTM-RNN and the last hidden state of the encoding LSTM-RNN to get compact representation of the documents. We also use a hidden size of 20 for the embedded GRU cell and LSTM in our Evidence Assembler. In addition to that, since Welbl et al. (2017) show the poor performance of TF-IDF model we drop the TF-IDF document retrieval procedure and supervision at the first hop of the Document Explorer (with the document having highest TF-IDF score to query subject). We train all modules of our system jointly using Adam Optimizer (Kingma and Ba, 2014) with an initial learning rate of 0.001 and a batch size of 10. We also use a dropout rate of 0.2 in all our linear projection layers, encoding LSTM-RNN and character CNNs.