Project Report : CS 4803/7643 Spring 2020 - Multihop Question Answering

Arvindkumar Krishnakumar
Georgia Tech
akrishna@gatech.edu

Peter K McAughan
Georgia Tech
pmcaughan6@gatech.edu

Mitchell R Donley
Georgia Tech
mdonley3@gatech.edu

Abstract

This project aims at reproducing experiments for two published papers and verifying their validity and conclusions for the task of multi-hop question answering. The models in consideration for this analysis are **DecompRC**[9] *and* **Graph-based Recurrent Retriever**[1]. *In addition to reproduction, we also experiment with combining the merits of the two models to evaluate an alternative approach to solving the same task and present our results on the* **HotpotQA**[11] *dataset. In reproducing these experiments, we achieved very similar results as those claimed in their work. In our proposed combined approach we integrate a recurrent information retrieval system with question decomposition to provide a reading comprehension model that answers one single hop question at a time and utilizes a smaller, more accurate context. This resulted in improved results compared to DecompRC, but was unable to achieve similar results as the Graph-based Recurrent Retriever. With this we believe that having a narrow and accurate context is important and that question decomposition can be a useful idea, but there is more work need to achieve state of the art results.*

Figure 1: Example of a multi-hop question.

1. Motivation

Open domain question answering is a task that involves automatically answering questions across domains by retrieving their answers across a large collection of source documents. Multi-hop question answering is a sub-task that focuses on retrieving answers to questions, for which the evidence is spread across multiple documents and may or may not have semantic and lexical overlap with the original question itself. This task involves being able to analyze multiple documents in the context of each other, and an example of such a question is shown in Figure 1.

This poses a particularly hard problem for models such as simple non-parameteric approaches that use term frequencies like TF-IDF and BM25. While effective for simple single-hop questions, these models fail to extract answers in the multi-hop scenario. This is clearly shown through adversarial tests on single-hop reading comprehension models that prove there is a significant gap between machine and human reading comprehension levels [7].

Open domain multi-hop QA addresses this difference by using multiple steps to answer a single question without any given context for the reader. An example of such a question is ”What government position was held by the woman who portrayed Corliss Archer in the film Kiss and Tell?”, which requires multiple steps by first requiring the answer to ”Who portrayed Corliss Archer in the film Kiss and Tell?” (question one) and using the answer to question one, we need to answer ”What government position was held by [ANSWER]?” where [ANSWER] is the answer to question one. The context for these questions would be all the articles from Wikipedia. This presents two clear challenges: retrieving the correct contextual information and finding the multi-hop answer from the retrieved context.

Question Type	Question	Answer
Comparison	Were Scott Derrickson and Ed Wood of the same nationality?	yes
Bridge	2014 S/S is the debut album of a South Korean boy group that was formed by who?	YG Entertainment

Table 1: HotpotQA Examples

2. Related Work

Retrieve-and-Read: Chen et al. [2] developed a "retrieve and read" approach which separates open domain question answering into a two part process: information retrieval and reading comprehension. This allows the two models to be trained and developed independently. They employ a TF-IDF based document retriever along with a state-of-the-art reading comprehension model. While this allows optimizing each part individually, Qi et al. and Ding et al. [10, 5] argue that to answer multi-hop questions, these two processes need to be jointly trained.

Graph-based retrieval: Ding et al. present CogQA [5] which comprises two systems that iteratively arrive at a final reasoning path and answer and is inspired by the observed method used by the human brain. The first system extracts entities and answer candidates that are relevant to the question and organizes them as a cognitive graph. The second system reasons over this graph, and finds "clues" that guide the first system to extract better entities. This process is repeated until all possible answers are found, and then the final answer is derived from reasoning results. This approach was notably successful, as achieved results significantly higher than other models such as BERT and DecompRC.

Iterative retrieval: Qi et al. develop a method "GoldEn Retriever" that works similar to CogQA but instead of using a graph based document structure, GoldEn uses the question and all currently available supporting facts from earlier queries to generate new queries that encode all the context learned so far. More recently Fang et al. [6] created a Hierarchical Graph Network (HGN) that utilized varying levels of granularity to create a more exact graph of the supporting facts where nodes are as broad as paragraphs and as fine grained as single entities. HGN optimizes over several sub-tasks to ensure that the answers are correct and explainable through the reasoning paths (paragraph selection, supporting fact prediction, entity prediction, and answer span extraction). Asai et al. [1] use a recurrent information retrieval system to develop the reasoning paths. On top of this, they maintain the top k reasoning paths and generate a set of answers as the final step, before choosing the best of these final answers.

Question Decomposition: One of the earlier approaches, Min et al. [9] focuses on decomposing the question to simplify the task of the reader compared to improving the information retrieval. DeCompRC attempts to split a multi-hop question into multiple single-hop questions of four different sub-types (bridging, intersection, comparison, one-hop). From here, the reader predicts the answer for each sub-question with the given context and selects the sub-question and answer with the max score as the final answer. See Figure 2 below, seen in the DecompRC paper [9], for an example of this decomposition and scoring process.

We believe that the analysis of systems that utilize all of the merits of these above approaches is lacking, and the results of integrating all these together is valuable information. Because open domain multi-hop question answering is complex, there are many sub-tasks that can be optimized to improve the results. The works focus on improving either the information retrieval, supporting fact prediction, question decomposition, or reading model, but none of the works attempt to utilize all these aspects. While combining some of these approaches would make the system more complex, each model would have to focus on a simpler sub-task.

Question answering systems are widely used in active areas of research as well as many places in industry. As we strive to implement models that are able to understand and generate language that is more grounded, performing well in multi-hop question answering is a crucial step to ensure models are understanding the different complexities in language and not searching for shortcuts. With this in mind, researchers are actively looking for new approaches that can improve language models to perform better at multi-hop question answering. With large performance gains, areas of work such as chat-bots, search queries, and knowledge graphs will lead to an overall improvement in the AI-Natural Language domain.

If we are successful, we will be able to show that not only is improving the retriever and reader a worthwhile approach, but that the structure and simplicity of the question is just as important. This will suggest that question decomposition is a useful sub-task in the process of jointly optimizing an open domain multi-hop question answering model.

2.1. Data

HotpotQA is the dataset we will be using as it contains natural, multi-hop questions, that were curated with strong supervision for supporting facts to enable more explainable question answering systems. It has been designed in a dis-

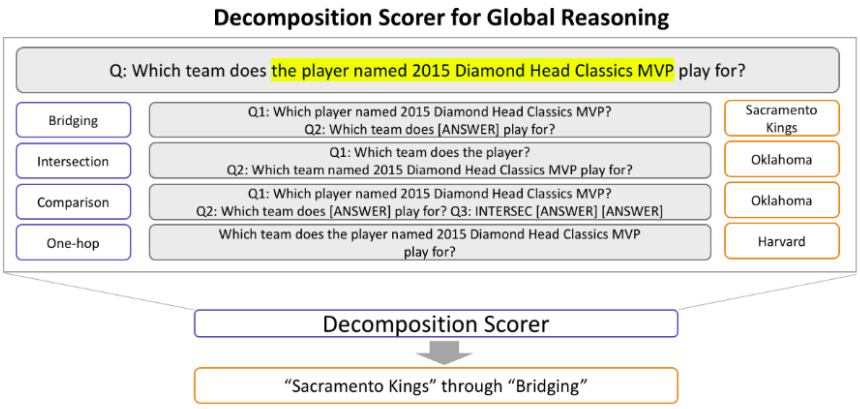

Figure 2: Question Decomposition Example

Question Type	Count
Bridge	78,909
Comparison	18,943

Table 2: Question Count

tractor setting as well as a full Wikipedia setting, containing 90,500 questions for training and 7,400 for validation [11]. The distractor settings gives 10 paragraphs with eight being distractions and two containing relevant information. This introduces the task of narrowing down the provided documents to those that contain relevant facts, and can make the task much more difficult. The full wikipedia setting gives a dump of English Wikipedia from 2018 to use for finding the correct answer and supporting facts. This data dump contains 5.7 million articles, of which only a few will pertain to any particular question in the dataset. Because we don't have access to ground truth values of the test set of HotpotQA, we will base our discussions on the validation set and compare them to those presented in the papers. Table 1 shows the number and type of HotpotQA questions, and Table 2 gives examples of different questions and answers.

3. Multihop Question Answering

Our initial goal in this project was to reproduce the results and experiments presented in the ICLR 2020 paper submission by Asai et al. titled *"Learning to Retrieve Reasoning Paths over Wikipedia Graph for Question Answering"* available here. To increase the scope of the project, we also chose to reproduce the results of the ACL 2020 paper by Min et al. titled *"Multi-hop Reading Comprehension through Question Decomposition and Rescoring"* available here. We have also presented our own approach to solving the multi-hop question answering task by utilizing the merits of the above techniques.

3.1. Reasoning Paths Retrieval Reproducibility

The authors use Wikipedia for open-domain QA, where each article is divided into paragraphs. Each paragraph is considered as the retrieval target. Given a question, the framework aims at deriving the answer by retrieving and reading reasoning paths, each of which is represented with a sequence of paragraphs. The authors formulate the task by decomposing the objective into the retriever objective that selects reasoning paths relevant to the question, and the reader objective that finds the answer.

Graph Construction: This work proposes document retrieval as a graph-based neural path search over Wikipedia articles where each paragraph represents a node and each internal hyperlink is considered as an edge. They use this approach mainly because evidence paragraphs for a complex question do not necessarily have lexical overlaps with the question, and entities in the graph usually entail other paragraphs as well.

Recurrent Retriever: The graph-based recurrent retriever iteratively retrieves evidence documents for multiple reasoning paths conditioned on the history of documents that were retrieved previously. The RNN-based module first brings in relevant seed nodes and uses this as the starting step for estimating sequences of paragraphs that could contain the answer to the question. This is done based on BERT's [CLS] token representations on independently encoded question and candidate paragraphs. The process terminates on encountering an end-of-evidence symbol to allow capturing arbitrary length reasoning paths unlike previous methods that had the search termination conditions hard-coded.

The authors use annotated evidence paragraphs in a supervised manner with a "negative sampling + data augmentation and inference-time decoding" strategy to train the re-

triever. Ground-truth reasoning paths were derived using the annotated data in each dataset. The negative samples then help the retriever discriminate between relevant and irrelevant paragraphs at each step. Additional reasoning paths have also been added to the training data to stabilize the training process. For the sequential prediction task, they jointly optimize the model and BERT using a binary cross-entropy loss function.

Reader: The reader model verifies the reasoning paths and returns an answer span from the most plausible reasoning path. The reader is modeled for multi-task learning of reading comprehension and reasoning path re-ranking. For the reading comprehension task, they use BERT [4], where the input is the concatenation of the question text and the text of all the paragraphs in the path. This allows the reader model to leverage the self-attention mechanism across concatenated paragraphs. They reuse the same model for re-ranking.

The multi-task reader model is trained using ground-truth evidence paragraphs that were used for training the retriever. Negative samples are added to training data to help the model discriminate between relevant and irrelevant paragraphs. The objective function used is the sum of cross entropy losses for the span prediction and re-ranking tasks.

The paper also uses search strategies like beam search, to reduce computational complexity of the search tree and help the framework work better on both single and multihop questions.

Reproducing this paper involved understanding the authors' approach to solving this problem, and experimenting specifically with different modules in their implementation, before adapting them to our specific use-case. The authors have implemented their method in PyTorch. More details on the experiments performed are in Section 4.1.

3.2. DeCompRC Reproducibility

When recreating the DecompRC paper, we utilized the open-source code repository described earlier. This repository was implemented in PyTorch and contained pre-trained models for these tasks. We utilized their pretrained models for question decomposition as well as question answering. These models decompose the question using span prediction with the assumption that sub questions can be extracted with light editing to make a complete sub-question. This is done using BERT to encode the question and a pointer matrix to determine scores for the start index where each type of sub question has a pre-defined length controlled by a hyperparameter. This final matrix Y denotes the span prediction start index scores where $Y_{ij} = \mathbb{P}(i = ind_j)$ gives the probability that the ith word is the jth index from the pointer and the product of the top c indices determine the span.

From here the reading comprehension model is used to answer each sub-question, giving a final result after the second sub-question is answered. This is done using BERT with a paragraph selection approach [3]. Reasoning across multiple paragraphs is difficult due to the paragraph selection, but this is somewhat alleviated by utilizing the sub-questions.

With the final answers to each decomposed sub-question type (bridge, intersection, comparison, and onehop), a decomposition scorer is used to determine which question decomposition, answer pair is the most likely. This is determined using BERT to encode each question, question type, answer, and evidence set which provides an output U such that $U_t \in \mathbb{R}^{n \times h}$ where t denotes the decomposition type. The final score for each question decomposition is given by

$$p_t = sigmoid(W_2^T max(U_t))$$

where t is the decomposition type and W_2 is a trainable matrix. The max p_t is the chosen decomposition, question, answer, evidence set. A visual example of the scorer can be seen in Figure 1. Results for the reproducibility can be seen in Section 4.2.

3.3. Our Approach

In our approach, we combine three works. We incorporate these approaches as each improves upon a subtask in open-domain multi-hop question answering. Min et al. [9] develops a human level question decomposer which transforms a multi-hop question into two single-hop sub questions. Asai et al.[1] proposes a graph-based recurrent information retrieval system that accurately retrieves contextual information without large computational complexity. Lastly, Min et al. [8] presents a BERT-based single-hop reading comprehension model that performs well on multi-hop question answering. With the combination of these three models, we present a system that simplifies multi-hop question answering into simpler tasks and hope to improve performance, while reducing the complexity of any single model.

We combine these by first, obtaining the top-k reasoning paths using [1] where k is a hyperparameter. Once we have the top-k paths, we extract all unique context paragraphs which will replace the original context. From here, we compute the question decompositions and each sub-question is answered with the reading comprehension model from Min et al. where the final answer to each decomposition is the answer to the last sub-question. Each of these question, decomposition, context, and answer sets are scored and the set with the max score is chosen as the final answer. A visualization of this pipeline can be seen in Figure 3.

We chose this approach as it allows for modularity between question decomposition, scoring, and selecting reasoning paths. Although past works claim this is not always ideal. [1] has shown that this approach is still able to get

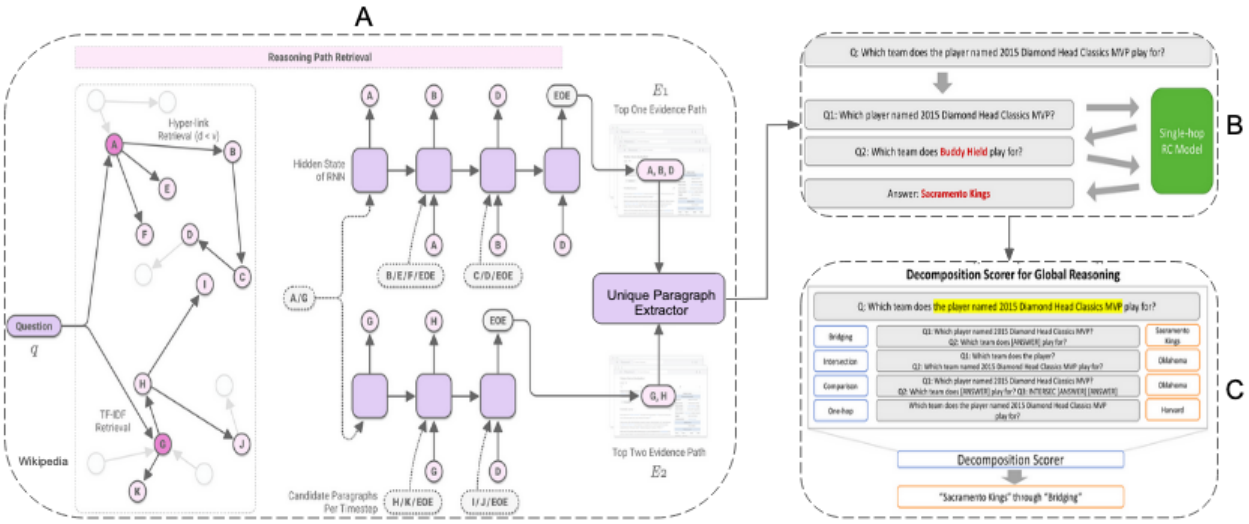

Figure 3: Our pipeline: Model A retrieves the reasoning paths and selects supporting facts. Model B is used to decompose the question into sub-questions and answer each decomposition. Model C scores and selects the final answer and decomposition type.

near state of the art. Our approach specifically attempts to address the issue of finding good supporting facts and answer the question accurately and our core assumption is that breaking down this process into simpler sub-questions will allow the system to perform better.

One major problem that could arise from our approach is that if any of these models in Figure 3 provide incorrect information then this information will continue to propagate into the final answer without having the ability to re-adjust the solution. This did end up being an issue both with the selection of reasoning paths and the question decomposition. While we were not able to completely resolve the issue, we improved the performance by allowing more than the top reasoning path and combining the paths, mentioned above.

4. Experiments and Results

4.1. Reasoning Path Retrieval

As we don't have access to ground truth labels for the HotpotQA test set, Table 3 shows the results of the Graph-based Recurrent Retriever on the HotpotQA fullwiki development dataset. We were able to replicate results that were presented by the authors in their paper. This involved computing TF-IDF representations of the fullwiki dataset. Due to computational limitations we had to restrict this to only introductory paragraphs, and this same decision was made in the oringinal paper itself. This resulted in a graph with about 5.2 million nodes and 23.4 million edges. Similar to the paper itself, we use the pre-trained BERT models from Devlin et al. [4] using the uncased base configuration (d = 768) for our retriever and the whole word masking uncased large (wwm) configuration (d = 1024)for our readers. We use the same hyperparameters for the TF-IDF based seed retrieval model as Chen et al. [2]. We used a beam size of 8 and the number of initial seed paragraphs for TF-IDF as 500, as these were hyperparameters chosen by the authors using the HotpotQA Dev set.

We were able to successfully replicate the results of the paper on the HotpotQA fullwiki development set in terms of both the Exact-Match score and the F1 score as can be seen in Table 3. The exact match score defines the percentage of the predictions that match any one of the ground truth answers exactly. The F1 score of the model is calculated as (2 x Precision x Recall) / (Precision + Recall), where precision is defined by the number overlaps between the prediction and ground truth divided by the total number of predictions, while the recall is calculated by the overlaps divided by the number of ground truth answers. For HotpotQA, we are also able to match Supporting Fact F1 (SP F1) and Supporting Fact EM (SP EM) to evaluate the sentence-level supporting fact retrieval accuracy.

Another experiment that was conducted was to verify the importance of beam search in reasoning path retrieval. Through an ablation experiment that was performed by using a greedy strategy instead of a beam search based retrieval, we noticed a drop in both the F1 and EM scores as can be seen in Table 3. However, we noticed slightly different numbers than those reported by the authors while being able to replicate the exact same environment settings used by them. We will be raising a pull request to address this

Model	QA		SP	
	EM	F1	EM	F1
Graph-based Recurrent Retriever	60.5	73.3	49.1	76.05
Graph-based Recurrent Retriever (Ablation)	56.8	69.4	46.4	74.05
DecompRC	30.00	40.65	N/A	N/A
Our Approach (top1-context)	49.43	63.31	N/A	N/A
Our Approach (top3-context)	49.35	63.3	N/A	N/A
Our Approach (top6-context)	47.24	60.82	N/A	N/A
Our Approach (all-context)	45.52	58.74	N/A	N/A

Table 3: F1 scores and Exact Matches reported on the Dev set of HotpotQA full wiki setting.

issue shortly.

Here are some strengths of this approach:

- Their reasoning path retrieval far outperforms most other existing retrieval approaches on the Hotpot full-wiki dataset. One important factor for this is that existing iterative retrieval methods fix the number of reasoning steps while this accommodates arbitrary steps of reasoning.
- This paper also emphasizes the use of layer normalization in conjunction with the BERT outputs in obtaining an end-of-evidence symbol during the retrieval phase.
- By independently encoding the paragraphs with the question, the retriever is made scalable, whereas the reader jointly learns to predict the significance of a reasoning path and answer the question, and thus leverages the self-attention mechanism in the retrieved reasoning paths.
- They also train the recurrent retriever by using the training examples for the supporting fact prediction task. This is accomplished by substituting the question-paragraph encoding with a question-answer-sentence encoding.
- Bridge entities are rarely stated clearly in bridge questions. This makes it harder to discover the paragraphs entailed by the bridge entities, but this approach still outperforms other existing methods by a good margin in this scenario.
- Comparison questions can usually be answered from a single paragraph, and thus the model selects only one paragraph for these questions, resulting in lower scores on the comparison-type questions.

4.2. DecompRC

For the reproduction of the DecompRC paper, Table 4 shows the results given on the leaderboard of HotpotQA distractor setting compared to the results we obtained from our

Model	QA	
	EM	F1
DeCompRC (reproduced)	55.11	70.01
DeCompRC	59.20	69.63

Table 4: F1 scores and Exact Matches reported on the Dev set of HotpotQA distractor setting for DecompRC.

local reproduction. Although we reused their pre-trained models, we achieved slightly better F_1 performance and worse EM performance. We believe this is due to the changes in data from HotpotQA. When we attempted to reproduce DeCompRC, we used a more recent version (v1.1) of the HotpotQA dataset which removed extremely short questions. We originally hypothesized this would decrease performance, but contrary to our expectation, these results were better. We believe that removing extremely short questions removed samples that had overall bad decomposition scores, and thus the model was able to act more successfully.

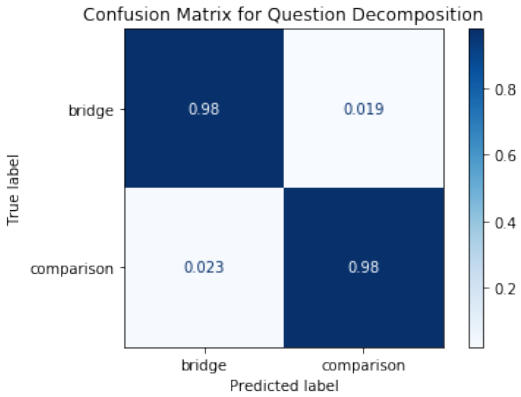

Figure 4: Confusion matrix of question decomposition.

An interesting note that we see, is that the decomposition model is incredibly accurate at choosing the decomposition.

In Figure 4, we see that the model is performing at almost perfect accuracy for question decomposition. To convert the 4 possible decompositions in DeCompRC (bridge, intersection, comparison, onehop) to the 2 question types of HotpotQA (bridge, comparison), we ignored all questions that were labelled onehop and combined all intersection and bridge type questions because when combined these two categories covered 98% of the HotpotQA bridge questions. Although this does simplify the problem, it still shows that accurately performing decomposition is a feasible task.

On the other hand, the difficult decompositions could be categorized as onehop by DeCompRC as the onehop type was predicted 28.4% of the time in the development dataset. With looking into this further, the F_1 and EM for each decomposition type can be seen in Table 5 where we see onehop questions are performing at about the same level as the other three decomposition types. Thus, we cannot prove that difficult questions are failing to be decomposed, but something we can see from this is that nearly 30% of the questions are being answered as a one hop question and achieving similar results to the other decomposition types. These findings show that while some multi-hop questions are not as difficult as expected and decomposition may not be necessary, there is a larger number of questions that do benefit from decomposition and require a complex understanding of the question and context. Because of this, we attempt to improve these results with a better information retrieval system so the context is more accurate than that of the distractor settings. This is done with the approach mentioned in Section 3.3 and the results can be seen in Section 4.3.

Decomposition Type	**QA**	
	EM	F1
Intersec	55.68	72.96
Comparison	52.17	63.14
Bridge	60.16	75.42
Onehop	50.91	66.73

Table 5: F1 scores and Exact Matches reported on the development set of HotpotQA distractor setting for each question decomposition.

4.3. Our Approach

We utilized F1 and EM (exact match) scores as measures of success for our model, as these are the easiest way to compare our performance across many other models who have attempted this task on HotpotQA. The dataset holds a publicly available leaderboard where researchers can display their F1 and EM scores, as well as include a description of their approach. Min et al. [9] and Asai et al. [1] are both on this leaderboard, and thus we can compare our results straightforwardly with each individual model.

From our results, we can see that our combined approach performed significantly better than DecompRC, but worse than the Graph-Based Recurrent Retriever. This implies that information retrieval models that utilize a recurrent structure are a more successful approach than TF-IDF retrievers when utilizing reader models based on decomposition such as DecompRC. The architecture and forward pass of an RNN is a more complex calculation than a TF-IDF calculation on a broad set of documents, and seems to be able to capture more context when extracting reasoning paths. Narrowing down the corpus of information to a relatively small, relevant portion helps DecompRC function at a higher performance, and we believe this is the case because the model has much less room for error if the original input is cut down.

This hypothesis is supported by the last rows of Table 3. When we only feed in the most likely reasoning path from the information retrieval process into DecompRC, we achieve the highest evaluation scores. As we relax our criteria for what information we put into DecompRC, however, we see that it performs worse and moves closer to the score of classic DecompRC with simple TF-IDF preprocessing. This seems to suggest that one shortcoming of DecompRC is its inability to realize what isn't important in a question context, and this can be remedied by supplanting a information retrieval model such as we have done in this project. Figure 5 displays a further breakdown of this performance differential to examine the performance on various question types. The "Top 1" reasoning path contains the top ranked collection of contexts that form a reasoning path, while the "Top N" reasoning path contains all the unique contexts from the top N ranked reasoning paths.

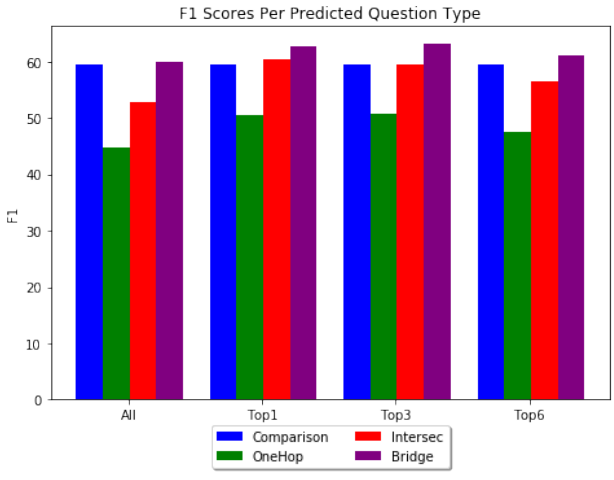

Figure 5: F1 Scores for Question Type

We see that providing more information to DecompRC

Student Name	Contributed Aspects	Details
Arvind	Graph-based Recurrent Retrieval	Reproducability, Experiments and Ablation Study.
Peter	Combined Approach	Information Pipeline from Information Retriever to DecompRC.
Mitchell	DeCompRC and Visualizations	Reproducibility: DecompRC, Analyzed decompositions types

Table 6: Contributions of team members.

doesn't seem to have an impact on the performance of Comparison questions, but the ability to answer one-hop, bridging, and intersection question falls steadily as provided information increases. This makes sense, as comparison questions are often less open-ended than the other question types. For example, the comparison question "Which country has more languages, India or China?" provides much more structure and a smaller range of reasonable answers than a question such as "Which NBA player led the league in points scored in 1991?". It seems that DecompRC is able to extract comparison context in a way that scales well with information size, but less so for intersection, bridging, or one-hop questions.

In Table 3, we still see that the model with the highest evaluation scores is the original Graph-based Recurrent Retriever created in the publication by Ding et al. This implies that for the task of multi-hop question answering, the approach of using a decomposer to break down a multi-hop question into single-hop questions might be inferior to simply using a model that can handle multi-hop questions, even when using an effective information retrieval system for obtaining reasoning paths.

5. Conclusion

Thus, through this project, we were able to successfully reproduce the experiments and results from two papers, **Reasoning Path Retrieval** and **DecompRC**, and understand the techniques behind current state-of-the-art approaches to multihop question answering. Although our results differ slightly from the reported results of these papers, we provide reasonable justifications for why this might be the case. We were also able to leverage modules from these approaches to build our own system that was able to improve upon the results from DecompRC. This emphasises the importance of using an approach that addresses all components in order to solve complex tasks, allowing scalability and reusability of certain modules within the pipeline that can be trained together or separately, to produce better results.

5.1. Future Work

Below are a few ideas that we believe could be implemented to improve performance.

1. Utilize each of the top-k reasoning paths independently as contexts for the question decomposition and reading comprehension. This would maintain a small and hopefully accurate context while allowing the system to explore multiple different reasoning paths and choose the best one after decomposition.

2. Switch reading comprehension models to the model used by Asai et al. in *Retrieving Reasoning Paths*. Theirs performed very well without any question decomposition.

3. Improve the categories of the question decomposition so that onehop categorization is minimized as much as possible

4. Implement a hierarchical retrieval system so paragraphs, sentences, or entities can be used as the context instead of only paragraphs. This is similar to [6].

5. Determine a method to jointly train all of these tasks as one large task because Chen et al. [2] mentioned this can be an issue.

6. Work Division

Delegation of work among team members has been provided in Table 6.

References

[1] Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. Learning to retrieve reasoning paths over wikipedia graph for question answering. *arXiv preprint arXiv:1911.10470*, 2019. 1, 2, 4, 7

[2] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions, 2017. 2, 5, 8

[3] Christopher Clark and Matt Gardner. Simple and effective multi-paragraph reading comprehension, 2017. 4

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 4, 5

[5] Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. Cognitive graph for multi-hop reading comprehension at scale. *arXiv preprint arXiv:1905.05460*, 2019. 2

[6] Yuwei Fang, Siqi Sun, Zhe Gan, Rohit Pillai, Shuohang Wang, and Jingjing Liu. Hierarchical graph network for multi-hop question answering, 2019. 2, 8

[7] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems, 2017. 1

[8] Sewon Min, Eric Wallace, Sameer Singh, Matt Gardner, Hannaneh Hajishirzi, and Luke Zettlemoyer. Compositional questions do not necessitate multi-hop reasoning. *arXiv preprint arXiv:1906.02900*, 2019. 4

[9] Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. Multi-hop reading comprehension through question decomposition and rescoring. *arXiv preprint arXiv:1906.02916*, 2019. 1, 2, 4, 7

[10] Peng Qi, Xiaowen Lin, Leo Mehr, Zijian Wang, and Christopher D. Manning. Answering complex open-domain questions through iterative query generation, 2019. 2

[11] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018. 1, 3