arXiv:2004.02995v1 [cs.CL] 6 Apr 2020

Multi-Step Inference for Reasoning Over Paragraphs

Jiangming Liu[*]
ILCC, University of Edinburgh
Jiangming.Liu@ed.ac.uk

Matt Gardner
Allen Institute for AI
mattg@allenai.org

Abstract

Complex reasoning over text requires understanding and chaining together free-form predicates and logical connectives. Prior work has largely tried to do this either symbolically or with black-box transformers. We present a middle ground between these two extremes: a compositional model reminiscent of neural module networks that can perform chained logical reasoning. This model first finds relevant sentences in the context and then chains them together using neural modules. Our model gives significant performance improvements (up to 29% relative error reduction when combined with a reranker) on ROPES, a recently-introduced complex reasoning dataset.

1 Introduction

Performing chained inference over natural language text is a long-standing goal in artificial intelligence (Grosz et al., 1986; Reddy, 2003). This kind of inference requires understanding how natural language statements fit together in a way that permits drawing conclusions. This is very challenging without a formal model of the semantics underlying the text, and when polarity needs to be tracked across many statements.

For instance, consider the example in Figure 1 from ROPES (Lin et al., 2019), a recently released reading comprehension dataset that requires applying information contained in a background paragraph to a new situation. To answer the question, one must associate each category of flowers with a polarity for having brightly colored petals, which must be done by going through the information about pollinators given in the situation and linking it to what was said about pollinators and brightly colored petals in the background paragraph, along with tracking the polarity of those statements.

[*]Work during an internship with AI2.

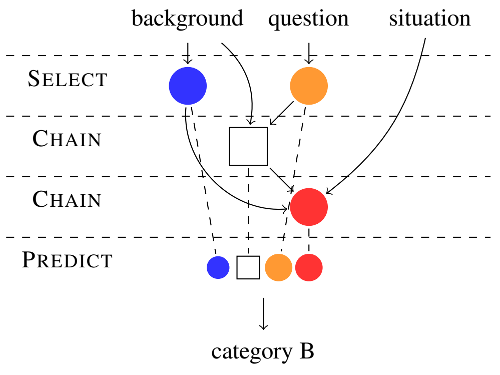
Background: Scientists think that the earliest flowers attracted *insects and other animals, which spread pollen* from flower to flower. *This greatly increased the efficiency of fertilization over wind-spread pollen*, which might or might not actually land on another flower. *To take better advantage of* $\boxed{\textit{this animal labor}}$, *plants evolved traits such as* $\boxed{\textit{brightly colored petals to attract pollinators.}}$ In exchange for pollination, flowers gave the pollinators nectar.

Situation: Last week, John visited the national park near his city. He saw many flowers. His guide explained him that there are two categories of flowers, category A and category B. *Category A flowers spread pollen via wind, and category B flowers spread pollen via animals.*

Question: Which category of flowers would be *more likely to have brightly colored petals*?

Answer: category B

(a)



(b)

Figure 1: (a) An example in ROPES; (b) the chained reasoning that our model performs on the example. The model first (softly) selects relevant parts of the background and question, then successively chains them, making a prediction after including the situation in the chaining.

Prior work addressing this problem has largely either used symbolic reasoning, such as markov logic networks (Khot et al., 2015) and integer linear programming (Khashabi et al., 2016), or black-box neural networks (Jiang et al., 2019; Jiang and Bansal, 2019). Symbolic methods give some measure of interpretability and the ability to handle logical operators to track polarity, but they are brittle, not able to handle the variability of language. Neural networks often perform better on practical datasets, as they are more robust to paraphrase, but they lack any explicit notion of reasoning and are hard to interpret.

We present a model that is a middle ground between these two approaches: a compositional model reminiscent of neural module networks that can perform chained logical reasoning. The proposed model is able to understand and chain together free-form predicates and logical connectives. The proposed model is inspired by neural module networks (NMNs), which were proposed for visual question answering (Andreas et al., 2016b,a). NMNs assemble a network from a collection of specialized modules where each module performs some learnable function, such as locating a question word in an image, or recognizing relationships between objects in the image. The modules are composed together specific to what is asked in the question, then executed to obtain an answer. Similarly, we design modules that are targeted at the reasoning necessary for ROPES and compose them together to answer questions.

We design three kinds of basic modules to learn the neuro-symbolic multi-step inference over questions, situation and background passages. The first module is called SELECT, which determines which information (in the form of spans) is important to the question; the second module is called CHAIN, which captures the interaction from multiple statements; the last one is called PREDICT, which assigns confidence scores to potential answers. The three basic modules can be instantiated separately and freely combined.

In this paper, we investigate one possible combination as our multi-step inference on ROPES. The results show that with the multi-step inference, the model achieves significant performance improvement. Furthermore, when combined with a reranking architecture, the model achieves a relative error reduction of 29% and 8% on the dev and test sets in the ROPES benchmark. As ROPES is a relatively new benchmark, we also present some analysis of the data, showing that the official dev set is likely better treated as an in-domain test, while the official test set is more of an out-of-domain test set.

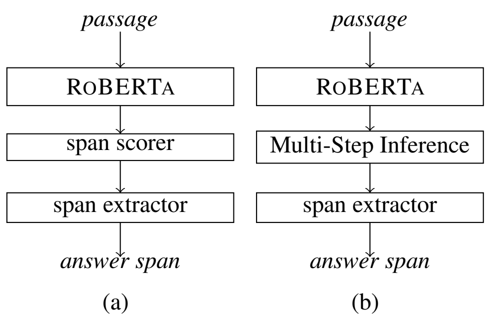


Figure 2: (a) The baseline system; (b) the proposed system with multi-step inference.

2 Systems

We first describe the baseline system, a typical QA span extractor built on ROBERTA (Liu et al., 2019), and then present the proposed system with multi-step inference, which is built on the top of baseline system. Furthermore, we introduce a reranker with multi-step inference given the output of the baseline system.

2.1 Baseline

Our baseline system is a span extractor built on the top of ROBERTA, which is shown in Figure 2(a). Two scores are generated for each token by span scorer, showing the chance to be the start and the end of the answer span:

$$\bar{s}_k, \bar{e}_k = \text{qa\_score}(x_k),$$

where x_k is the representation of kth ROBERTA token, $\bar{s}_k$ and $\bar{e}_k$ are the scores of the start and the end of answer spans, respectively, and qa_score$(\cdot)$ is a linear function. The span with highest start and end scores is extracted as the answer by span extractor:

$$\begin{aligned}[s_0, s_1, ..., s_n] &= \text{softmax}([\bar{s}_0, \bar{s}_1, ..., \bar{s}_n]) \\ [e_0, e_1, ..., e_n] &= \text{softmax}([\bar{e}_0, \bar{e}_1, ..., \bar{e}_n]) \\ i, j &= \text{argmax}_{i,j}\ s_i + e_j \quad (0 \le i \le j \le n),\end{aligned}$$

where the span$_{i,j}$ is the answer. In order to distinguish background passage B, situation passage S and questions Q, we concatenate the three passages with two special determiners "S:" and "Q:" to be a long passage "B S: S Q: Q", where the background and situation passage are regarded as

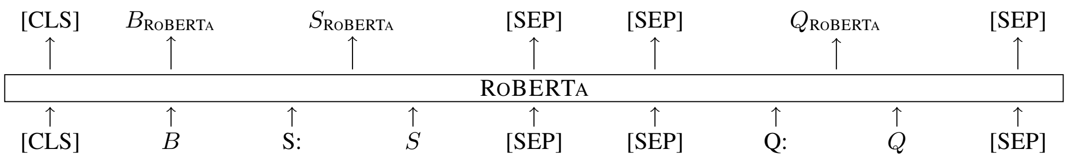


Figure 3: The RoBERTa component

the first segment and the question is the second segment in RoBERTa, which is shown in Figure 3.

2.2 Multi-Step System

Our proposed multi-step system is the baseline system equipped with multi-step inference. The architecture of the system is shown in Figure 2(b), where the multi-step system replaces the qa_score() function from the baseline with a series of neural modules targeted at chained inference, which outputs two scores for each token x_k:

$$\bar{s}_k, \bar{e}_k = \text{MS-Inference}(x_k, B_{\text{RoBERTa}}, S_{\text{RoBERTa}}, Q_{\text{RoBERTa}}),$$

where MS-Inference($\cdot$) is the multi-step inference model consist of several modules. These modules SELECT relevant information from parts of the passage, CHAIN the selected text together, then PREDICT the answer to the question given the result of the chaining.

As most of the questions in ROPES require the same basic reasoning steps, we use a fixed combination of these modules to answer every question, instead of trying to predict the module layout for each question, as done in prior work (Hu et al., 2017). This combination is shown in Figure 4: we separately SELECT important parts of the background passage and the question, then CHAIN them together to find a likely part of the background that will connect to the situation. Then we CHAIN that result with the situation, and finally PREDICT an answer, which is most often found in the situation text.

The actual operations performed by each of these modules is described below.

SELECT The select module, i.e. y = SELECT(X), aims to get the single representation given a sequence of token representations. Here, we use the weighted sum over the sequence of representations. Take $X = x_0, x_1, ..., x_n$, where x_k $(0 \leq k \leq n)$ have the same dimension, i.e. $x_k \in R^{1\times D_x}$ as the input and n is the length

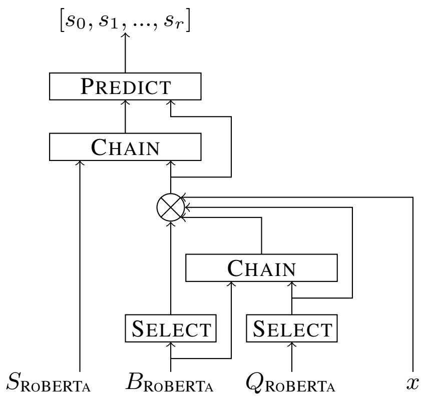


Figure 4: Multi-step inference model

of the input sequence:

$$w_k = f(x_k) \quad (0 \leq k \leq n)$$
$$[a_0, a_1, ..., a_n] = \text{softmax}([w_0, w_1, ..., w_n])$$
$$y = \sum_{k=0}^{n} a_k \cdot x_k,$$

where $f(\cdot)$ is a linear function.

CHAIN The chain module, i.e. y = CHAIN(X, z), aims to get the interaction representation given a set of representations $x_0, x_1, ..., x_n$, where x_k $(0 \leq k \leq n)$ have different dimensions, i.e. $x_k \in R^{1\times D_k}$ and a sequence of representations z where $z \in R^{1\times D_z}$. Here we take multi-head attention:

$$q = g([x_0; x_1; ...; x_n])$$
$$y = \text{attention}(q, z, z),$$

where $g(\cdot) : R^{1\times(D_0+D_1+...+D_n)} \Rightarrow R^{1\times D_z}$ is a linear function, ; means the concatenation of the vectors and attention($\cdot$) is instantialized with the multi-head attention:

$$\text{attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

where d_k is the dimension of K.

PREDICT The predict module, i.e. s = PREDICT(Y, r), aims to get the r scores given the reasoning procedure. The reasoning procedure is represented as the combination of outputs from all modules $y_0, y_1, ..., y_l$, where y_k $(0 \leq k \leq l)$ could have different dimensions, i.e. $y_k \in R^{1 \times D_k}$ and l is the number of modules and y_l. Here, we simply concatenate outputs of all modules to be a single vector:

$$s = \text{score}([y_0; y_1; ...; y_l]),$$

where $\text{score}(\cdot) : R^{1 \times (D_0 + D_1 + ... + D_l)} \Rightarrow R^r$ is a linear function, and $r = 2$ if the module is used to extract spans, e.g. the system requires start score and end score for each token, while $r = 1$ if the module is used to score candidates for reranker, which is described in the later Section.

2.3 Multi-Step Reranker

Finally, we propose a reranker with the multi-step inference model, aiming to choose the best answer from several candidate spans given by the baseline model, where a sampler is used to get candidates from the baseline system, and each candidate is represented as a vector by the span model. The multi-step inference model, slightly modified from above to take candidate spans instead of single tokens, produces a distribution over the candidates as potential answers, instead of over all possible spans in the passage.

Candidate sampling The baseline system outputs the spans with their scores and positions of the RoBERTa sequences. We sample the candidates with top c scores: $C = \{(i_0, j_0), (i_1, j_1), ..., (i_{c-1}, j_{c-1})\}$.

Span model To feed the candidate spans into our multi-step inference model, we represent each span as a vector by the end-point method. The span representation $x_{(i,j)}$ is the concatenation of the start token representations and the end token representations, i.e. $x_{(i,j)} = [x_i; x_j]$.

Instead of two scores to be start and end of the answer span for each token, the multi-step inference model in the reranker outputs one score for each span candidate:

$$\bar{o}_{(i,j)} = \text{MS-Inference}(x_{(i,j)}, B_{\text{RoBERTa}}, S_{\text{RoBERTa}}, Q_{\text{RoBERTa}}),$$

and then the candidate span with the highest score is chosen as the final answer:

$$[o_{(i_0,j_0)}, ..., o_{(i_{c-1},j_{c-1})}] = \text{softmax}([\bar{o}_{(i_0,j_0)}, ..., \bar{o}_{(i_{c-1},j_{c-1})}])$$

$$i, j = \text{argmax}_{i,j} o_{(i,j)} \quad (i,j) \in C.$$

Ensemble We take the ensemble strategy for the reranker. We train several rerankers, and build a voting system where each reranker makes a vote to the candidate to be the best answer. The candidate with the most votes is chosen the best answer through the voting system.

3 Data

We experiment with ROPES (Lin et al., 2019), a recently proposed dataset which focuses on complex reasoning over paragraphs for document comprehension. We noticed a very severe drop in performance between the ROPES dev and test sets during initial experiments, and we performed an analysis of the data to figure out the cause. ROPES used an annotator split to separate the train, dev, and test sets in order to avoid annotator bias (Geva et al., 2019), but this led to a large distributional shift between train/dev and test, as we show in this section.

3.1 Answer Constituent

Our analysis is based on looking at the syntactic category of the answer phrase. We use the syntactic parser (Kitaev and Klein, 2018) to obtain constituent trees for the passages in ROPES. The passages could have more than one answer span, and we assume that the last one is the answer. Given the syntactic structure and the answer span of the question, we take the constituent label of the first subtree that covers the answer span in bottom-up manner as the question type.

The four most frequent question types in ROPES are noun phrase (NP), verb phrase (VP), adjective phrase (ADJP) and adverb phrase (ADVP). Table 1 shows the examples for each type. Most of the answers to NP questions come from the situation, while the answers to the questions in other types come from the question. Most of these questions require the model to select between two obvious answer candidates.

3.2 Bias

We classify the questions on ROPES, and the statistics are shown in Table 2. We found that the distribution over question types in train set is similar to development set, where most of the question are NP type (85%) and the second frequent questions are ADJP type. However, the test set has a very different distribution over question

Type	Passage
NP	...The child poured two spoonfuls of sugar into cup A and three spoonfuls of sugar into **cup B**... Which cup has a higher concentration of sugar ?
	...They labeled it as **plant B** . They wanted to find out what makes a plant drought-resistant... In which plant there would be more water loss ?
VP	...In test B he used higher concentration of reactants. Now, he needs to know about the science... Would test B **increase** or decrease the frequency...
	...induced higher respiration rate in sample A. Then he induced no respiration rate in sample B... make their own glucose or **acquire it from other organisms** ?
ADJP	... patient A and patient B. John found out that patient A had more LDL, but patient B had more HDL... B have higher or **lower** risk of heart attack than patient A?
	...visible light. He noted microwaves as case A, infrared as case B, and visible light as case C...Would case A have **longer** or shorter wavelengths than case B?
ADVP	...Sample A was a strong acid, and sample B was a weak acid. David needed to ...sample A lose a proton less or **more easily** than sample B?
	...There is only one ice cube left so she takes it out and sets it in the glass on the table. She then refills...in the ice cube moving closer together or **farther apart** ?
Others	...Their mother takes them to see a doctor and to have their testosterone tested. The tests reveal that...Will Jimothy finish his growth spurt before or **after** Dwight?
	...He cut down on how much he eats every day and monitors his calorie intake, making sure that he is...Given Greg's BMI us 41, is he considered obese, **yes** or no?

Table 1: The examples in ROPES, where the bold red spans are answers.

types	train	dev	test
NP	84.17	85.19	47.19
VP	3.35	1.24	17.37
ADJP	9.20	10.25	19.36
ADVP	2.50	3.32	10.23
Others	0.78	0.00	5.85

Table 2: The percentage (%) of question types in ROPES.

types, where less than half of the questions are NP type, and there are more questions with VP, ADJP, ADVP and other types.

The different biases over questions in train, development and test raise challenges for reading comprehension systems; to perform well on test, the model must predict a significant number of answers from the question instead of from the situation, which only rarely happens in the training data. Given this distributional shift, it seems fair to characterize the official test as somewhat out-of-domain for the training data.

4 Experiments

In our experiments, we compare the performance of the systems we presented on ROPES, to investigate the performance of multi-step inference models.

	train	dev	test
# of backgrounds	513	51	171
# of situations	1,409	203	300
# of questions	10,924	1,688	1,710

Table 3: The ROPES dataset

4.1 Settings

Data We use the 10,924 questions as our training set, and 1,688 questions as dev set and 1,710 questions as test set, where each question has only one answer, which is a span from either the situation or the question. Table 3 shows the statistics on the ROPES benchmark. Due to the severe bias on question types in dev and test (seen in Section 3.2), we additionally set up an experiment using the dev set as an in-domain test set, by partitioning the training set into train (9,824 questions) and train-dev (1,100 questions).

Training Following the settings of prior work (Lin et al., 2019), we fine-tune the RoBERTa-large pre-trained transformer. The hidden sizes of all layers are set to 1024, and the number of heads on multi-step attentions is 8. All systems are trained in 1e-5 learning rate with 0.1 weight decay. We use the SGD training method with the batch size 8 and the Adam optimizer (Kingma and Ba, 2015).

Metrics Though ROPES was released using both exact match (EM) and F1 as metrics, we only report EM here, as F1 has been shown to correlate poorly with human judgments on ROPES (Chen et al., 2019a). F1 assumes that answers that share many overlapping words are likely similar; while this is largely true on SQuAD (Rajpurkar et al., 2016), where this particular F1 score was introduced, it is not true on ROPES, where things like *Village A* and *Village B* are both plausible answers to a question. All the systems are trained in three runs with different random seeds, and we post the average performance over the three runs.

4.2 Results

Table 4 shows the performance of the three systems. The multi-step system and multi-step reranker outperform the baseline system with 8.1% and 11.7% absolute EM accuracy on dev set, respectively, and with 2.4% and 2.0% EM accuracy on test set, respectively, showing that with multi-step inference, the system can achieve improvements. With the ensemble, the multi-step reranker performs best on dev and test sets.

As can be seen, the improvement of our model on the dev set is quite large. While performance is also better on the official test set, the gap is not nearly so large. To understand whether this was due to overfitting to the dev set or to the distributional shift mentioned in Section 3.2, Table 4 also shows the results on dev-test, our split that treats the official dev set as a held-out test set. Here, we still see large gains of 7.2% EM from our model, suggesting that it is indeed a distributional shift and not overfitting that is the cause of the difference in performance between the original dev and test sets. Properly handling the distributional shift in the ROPES test set is an interesting challenge for future work.

4.3 Analysis and Discussion

We conduct detailed analysis in this section, studying (1) the impact of various components of our model, (2) the gap between results on development and test set, (3) the strategy for sampling candidates for the reranker, and (4) the errors that the models cannot cover.

	dev	test	dev-test
baseline	59.7	55.4	56.2
multi-step	67.8	57.8	61.6
multi-step reranker	71.4	57.4	63.4
+ensemble	73.3	58.8	65.2

Table 4: The exact match scores by three systems.

	EM
multi-step	67.8
w/o Q SELECT	62.8 (-6.0)
w/o B CHAIN	62.3 (-6.5)
w/o B SELECT	65.9 (-1.9)
w/o S CHAIN	65.5 (-2.3)
average	(-3.7)
multi-step reranker	71.4
w/o Q SELECT	65.8 (-5.6)
w/o B CHAIN	67.7 (-3.7)
w/o B SELECT	64.9 (-6.5)
w/o S CHAIN	65.7 (-5.7)
average	(-5.4)

Table 5: The ablation results on development. Q SELECT denotes the question SELECT module; B CHAIN denotes the CHAIN module applied on the background and the question; B SELECT denotes the background SELECT module; S CHAIN denotes the CHAIN module applied on the situation and the previous chained reasoning.

Ablation Study We perform an ablation study on the multi-step system and the multi-step reranker. Table 5 shows the results on dev set by various ablated system. The performances of two systems drop down without any one module due to the property of the chained reasoning. The performance of the multi-step system without Q SELECT or B CHAIN drops (-5.3% EM) more than that of the multi-step system without B SELECT or S CHAIN (-2.1% EM). So Q SELECT module and B CHAIN play relatively more important roles. The performance of the multi-step reranker without Q SELECT, B SELECT or S CHAIN drops (-5.9% EM) more than that of the multi-step reranker without B CHAIN (-3.7% EM). The multi-step system drops average 3.7% EM while the multi-step reranker drops average 5.4% EM, showing that the multi-step reranker depends more on the modules.

	NP	VP	ADJP	ADVP
baseline	60.0	38.1	60.4	62.7
multi-step	68.8	39.7	61.3	72.6
multi-step reranker	71.8	38.1	63.8	75.0
+ensemble	75.0	42.9	61.3	78.6

Table 6: The exact match accuracy of most four frequent question types

	EM
10-fold	84.1
5-fold	82.4
2-fold	75.9
3-turn	59.9

Table 7: The average accuracy on training data for the multi-step reranker.

Answer Types We break down the overall accuracy by question type, which is shown in Table 6. All three systems perform substantially better on NP, ADJP, and ADVP questions than on VP questions. The main reason is that the VP questions are associated with complex and long answers, e.g. *acquire it from other organisms* or *make their own glucose*. The major improvements happen on answering NP and ADVP questions, which explains the gap between the scores on the development set, with large amount of NP questions, and the test set, with relatively more VP questions. The analysis can inspire the future work of investigating the specific inference programs for specific-type questions.

Candidate Sampling In order to train the reranker, we need training data with high-diversity candidates. However, the well-trained model hardly generates the similar candidates to the dev and test set, due to the overfitting to the training set. We investigate the various sampling strategies to get candidate answers which have the similar error distribution among train/dev and test set. We adopt four self-sampling methods, i.e. 10-fold, 5-fold, 2-fold and 3-turn. k-fold method means that the training data is partitioned into k parts, and $(k-1)$ parts are used to train a model which generates candidates answers for the other parts. k-turn method means that the training data is partitioned into k parts, and ith part is used to train a model which generates candidate answers for $(i+1)$th part.

Table 7 shows the average accuracy on training data. The accuracy on training data generate by k-fold self sampling method is very high, and they are not consistent with the dev and test set. The accuracy on training data generated by the 3-turn self sampling method is most similar to the accuracy on dev set (59.7% EM) and test set (55.4% EM) by the baseline system.

Table 8 shows the oracle of top k candidates on train, development and test set. Because oracle scores are the upper bound of the reranker, there is a trade-off that the upper bound is lower as fewer candidates are sampled, while the noise increases as more incorrect candidates are sampled. We found that top 3 provides a good trade-off for the reranker, where the oracle scores are 92.0% , 97.4% and 80.2% for train, development and test, respectively.

k	train	dev	test
1	59.9	59.7	55.4
2	81.4	64.8	61.9
3	92.0	97.4	80.2
4	93.8	98.3	83.6
5	94.9	98.7	85.9
10	96.1	99.4	88.5

Table 8: The oracle scores for top k candidates.

Error Analysis We analyze some errors that our proposed model made, aiming to discover the questions that our model could not cover. Table 9 shows some questions that our proposed model gives the incorrect answers. The questions require model to get the numeric information from the passage, and then compare the numeric relation (e.g. larger, smaller and equal) and target the effect of the relation in the background passage, where positive correlation between the prices and the sold number in example 1, positive correlation between the tolerance degree and usage times in example 2 and negative correlation between the crash rate the the number of cyclists in example 3. It seems that the model is not sensitive to the numeric information and their reasonings.

Also, the situations give more than two entities with their related information, and although the questions narrow down the multiple choices to two choices, the systems are still distracted by these question-irrelevant entities. The distraction come form the difficulty to associate the relevant information with the correct entities. The future work can be motivated by the discovery to design more

Example 1
Background: ... For many of the works, the price goes up as the edition sells out...
Situation: ...By the end of the week, they started to sell out. There were only 2 of the Mona Lisa,...,120 of The Kiss, 150 of The Arnolfini Portrait...
Question: Which limited edition most likely had it's price increased: The Kiss or Mona Lisa ?   **Answer:** The Kiss
Baseline: Mona Lisa   Multi-Step: Mona Lisa   Multi-Step Reranker: Mona Lisa
Example 2
Background: ...The tolerance for a drug goes up as one continues to use it after having a positive experience with a certain amount the first time...
Situation: ... Chris used it 12 times,...,Jimmy used it 42 times, Antonio used it 52 times, Danny used it 62 times, ...
Question: Who has a higher tolerance for roach: Jimmy or Antonio ?   **Answer:** Antonio
Baseline: Jimmy   Multi-Step: Jimmy   Multi-Step Reranker: Jimmy
Example 3
Background: ... That is to say, the crash rate per cyclist goes down as the cycle volume increases...
Situation: ...Day 1 had 500 cyclists left. Day 2 had 400 cyclists left. Day 3 had 300 cyclists left. Day 4 had 200 cyclists left....
Question: What day had a lower crash rate per cyclist: Day 1 or Day 2 ?   **Answer:** Day 1
Baseline: Day 2   Multi-Step: Day 2   Multi-Step Reranker: Day 2

Table 9: The examples of the answers to the questions by the baseline system, the multi-step system and the multi-step reranker.

modules to deal with this phenomenon.

5 Related Works

Neural Module Network The neural module network (NMN) was originally proposed for visual question answering tasks (Andreas et al., 2016b,a), and recently has been used on several reading comprehension tasks (Jiang et al., 2019; Jiang and Bansal, 2019), where they specialize the module functions such as FIND and COMPARE to retrieve the relevant entities with or without supervised signals. Instead, we generalize the modules with the attentions over the text and make these basic modules freely combinable.

Multi-Hop Reasoning There are several datasets constructed for multi-hop reasoning e.g. HOTPOTQA (Yang et al., 2018; Jiang et al., 2019; Jiang and Bansal, 2019; Min et al., 2019) and QANGAROO (Welbl et al., 2018; Chen et al., 2019b; Zhuang and Wang, 2019; Tu et al., 2019), which aims to get the answer across the documents. The term “multi-hop” reasoning on these datasets is similar to relative information retrieval, where one entity is bridged to another entity with one hop. Differently, the multi-step reasoning on ROPES aims to do reasoning over the effects of a passage (background and situation passage) and then give the answer to the question in the specific situation, without retrieval on the background passage.

Models beyond Pre-trained Transformer As the emergence of fully pre-trained transformer (Peters et al., 2018; Devlin et al., 2019; Liu et al., 2019; Radford et al.; Dai et al., 2019; Yang et al., 2019), most of NLP benchmarks got new state-of-the-art results by the models built beyond the pre-trained transformer on specific tasks (e.g. syntactic parsing, semantic parsing and GLUE) (Wang et al., 2018; Kitaev and Klein, 2018; Zhang et al., 2019; Tsai et al., 2019). Our work is in the same line to adopt the advantages of pre-trained transformer, which has already collected contextualized word representation from a large amount of data.

6 Conclusion

We propose a multi-step reading comprehension model that performs chained inference over natural language text. We have demonstrated that our model substantially outperforms prior work on ROPES, a challenging new reading comprehension dataset. We have additionally presented some analysis of ROPES that should inform future work on this dataset. While our model is not a neural module network, as our model uses a single fixed layout instead of different layouts per question, we believe there are enough similarities that future work could explore combining our modules with those used in other neural module networks over text, leading to a single model that could perform the necessary reasoning for multiple different datasets.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016a. Learning to compose neural networks for question answering. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1545–1554, San Diego, California. Association for Computational Linguistics.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016b. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48.

Anthony Chen, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019a. Evaluating question answering evaluation. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 119–124, Hong Kong, China. Association for Computational Linguistics.

Jifan Chen, Shih-ting Lin, and Greg Durrett. 2019b. Multi-hop question answering via reasoning chains. *arXiv preprint arXiv:1910.02610*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

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.

Mor Geva, Yoav Goldberg, and Jonathan Berant. 2019. Are we modeling the task or the annotator? an investigation of annotator bias in natural language understanding datasets.

Barbara J Grosz, Karen Sparck-Jones, and Bonnie Lynn Webber. 1986. Readings in natural language processing.

Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. 2017. Learning to reason: End-to-end module networks for visual question answering. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 804–813.

Yichen Jiang and Mohit Bansal. 2019. Self-assembling modular networks for interpretable multi-hop reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4464–4474, Hong Kong, China.

Yichen Jiang, Nitish Joshi, Yen-Chun Chen, and Mohit Bansal. 2019. Explore, propose, and assemble: An interpretable model for multi-hop reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2714–2725, Florence, Italy.

Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Peter Clark, Oren Etzioni, and Dan Roth. 2016. Question answering via integer programming over semi-structured knowledge. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 1145–1152. AAAI Press.

Tushar Khot, Niranjan Balasubramanian, Eric Gribkoff, Ashish Sabharwal, Peter Clark, and Oren Etzioni. 2015. Exploring markov logic networks for question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 685–694.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia. Association for Computational Linguistics.

Kevin Lin, Oyvind Tafjord, Peter Clark, and Matt Gardner. 2019. Reasoning over paragraph effects in situations. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 58–62.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Multi-hop reading comprehension through question decomposition and rescoring. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6097–6109, Florence, Italy. Association for Computational Linguistics.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke

Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Raj Reddy. 2003. Three open problems in ai. *Journal of the ACM (JACM)*, 50(1):83–86.

Henry Tsai, Jason Riesa, Melvin Johnson, Naveen Arivazhagan, Xin Li, and Amelia Archer. 2019. Small and practical bert models for sequence labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3623–3627.

Ming Tu, Guangtao Wang, Jing Huang, Yun Tang, Xiaodong He, and Bowen Zhou. 2019. Multi-hop reading comprehension across multiple documents by reasoning over heterogeneous graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2704–2713, Florence, Italy. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

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.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019. AMR parsing as sequence-to-graph transduction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 80–94, Florence, Italy. Association for Computational Linguistics.

Yimeng Zhuang and Huadong Wang. 2019. Token-level dynamic self-attention network for multi-passage reading In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2252–2262, Florence, Italy. Association for Computational Linguistics.