Multi-Step Inference for Reasoning Over Paragraphs

Jiangming Liu♠,*, **Matt Gardner**♣, **Shay B. Cohen**♠, **Mirella Lapata**♠
♠ ILCC, University of Edinburgh
♣ Allen Institute for AI
jiangming.liu@ed.ac.uk, mattg@allenai.org,
{scohen,mlap}@inf.ed.ac.uk

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.

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* | *this animal labor* |, *plants evolved traits such as* | *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*?
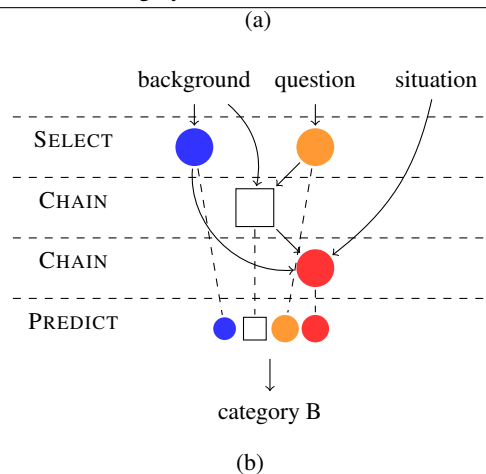**Answer:** category B

(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, unable 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. We design general 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, situations, and background passages. The first module is called SELECT, which determines which information (in the form of spans) is essential 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.[1]

2 Model

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. Furthermore, we introduce a reranker with multi-step inference given the output of the baseline system.

Following the standard usage of ROBERTA, we concatenate the background, the situation and question with two special determiners [S:] and [Q:] to be a long passage $P = $ [CLS] B [S:] S [SEP] [SEP] [Q:] Q [SEP], where the background B and situation S are regarded as the first segment and the question Q is the second segment, and [CLS] and [SEP] are the reserved tokens in ROBERTA.

2.1 Baseline

Our baseline system is a span extractor built on the top of ROBERTA. Given the passage representations from ROBERTA $P_{\text{ROBERTA}} = [x_0, ..., x_{n-1}]$, two scores are generated for each token by span scorer, showing the chance to be the start and the end of the answer span:

$$\bar{S}, \bar{E} = \text{QA\_score}(P_{\text{ROBERTA}}),$$

where $\bar{S} = [\bar{s}, \bar{s}_1, ...\bar{s}_{n-1}]$ and $\bar{E} = [\bar{e}_0, \bar{e}_1, ..., \bar{e}_{n-1}]$ $(0 \leq k < n)$[2] are the scores of the start and the end of answer spans, respectively. QA_score$(\cdot) : \mathbb{R}^{d_x} \Rightarrow \mathbb{R}^2$ is a linear function, where d_x is the output dimension of ROBERTA. The span with highest start and end scores is extracted as the answer by span extractor:

$$[s_0, s_1, ..., s_n] = \text{SOFTMAX}([\bar{s}_0, \bar{s}_1, ..., \bar{s}_n])$$
$$[e_0, e_1, ..., e_n] = \text{SOFTMAX}([\bar{e}_0, \bar{e}_1, ..., \bar{e}_n])$$
$$i^*, j^* = \arg\max_{i,j} s_i + e_j \quad (0 \leq i \leq j < n),$$

where the span$_{i^*,j^*}$ is the answer.

---

[1] Model code is available at https://github.com/LeonCrashCode/allennlp/blob/transf-exp1/allennlp/models/transformer_mc/roberta_models.py

[2] The answer spans always appear in the situation and question passage, so we mask the scores for the background passage.

2.2 Multi-Step Inference for ROPES

Instead of a simple span prediction head on top of ROBERTA, our proposed multi-step inference model uses a series of neural modules targeted at chained inference. Like the baseline, our model begins with encoded ROBERTA passage representations P_{ROBERTA}, but replaces the QA_score function with a MS-Inference function, which similarly outputs a span start and end score for each encoded token x_k:

$$\bar{S}, \bar{E} = \text{MS-Inference}(P_{\text{ROBERTA}})$$

The MS-Inference$(\cdot)$ function consists 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. These modules are applied on P_{ROBERTA} which is decomposed into $B_{\text{ROBERTA}}, S_{\text{ROBERTA}}, Q_{\text{ROBERTA}}$, denoting the token representations of ROBERTA for the background, the situation and the question, respectively.

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 was done in prior work (Hu et al., 2017). This combination is shown in Figure 2: we SELECT important parts of the question passage, and CHAIN them with the background passage to find a likely part of the background that supports answering the question (marked as red). Then we SELECT important parts of the background passage, which are combined with previous results that we have (marked as blue), and we CHAIN the combined information to find relevant parts of the situation passage (marked as green), and finally PREDICT an answer (marked as black), which is most often found in the situation text. The intuition for how these modules work together to piece together the information necessary to answer the question is shown in Figure 1. The actual operations performed by each of these modules is described below.

SELECT The select module, i.e. $z = \text{SELECT}(Y)$, where $Y \in \mathbb{R}^{n \times d_x}$ and $z \in \mathbb{R}^{d_x}$, aims to find the important parts of its input and summarize in a single vector. It first uses a learned linear scoring function, $f(\cdot) : \mathbb{R}^{d_x} \Rightarrow \mathbb{R}$, to determine which parts of its input are most important, then converts those scores into a probability
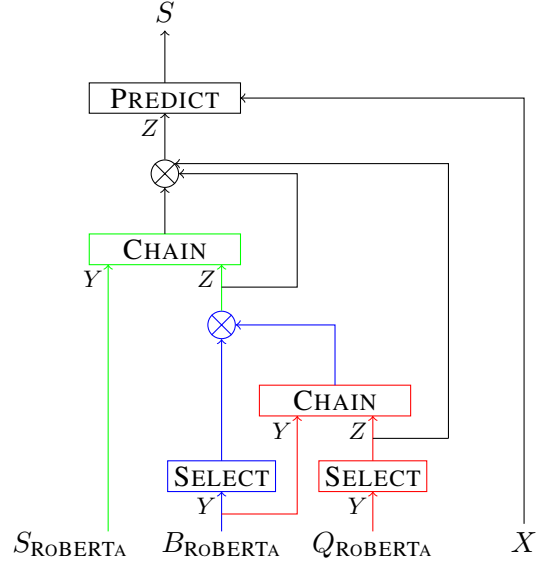

Figure 2: Multi-step inference model, where $\otimes$ is the operation to collect multiple vectors as a list, Z, Y are the interfaces of the modules, and X is a token representation to be scored as start/end of the answer span in the QA systems, or a candidate span representation to be scored in the reranking systems.

distribution using a SOFTMAX operation, and computes a weighted sum of the inputs:

$$W = f(Y)$$
$$A = \text{SOFTMAX}(W)$$
$$z = A^{\text{T}}Y,$$

CHAIN The chain module, i.e. $z = \text{CHAIN}(Y, Z)$, computes the interaction between an input matrix Y and a list of the input vectors $Z = [z_0, z_1, ..., z_{l-1}]$, where $Y \in \mathbb{R}^{n \times d_x}$, $z_k \in \mathbb{R}^{d_k}$ and d_k is the dimension of the kth input vector ($0 \le k < l$), and again outputs a summary vector of this interaction $z \in \mathbb{R}^{d_x}$. Intuitively, this module is supposed to chain together the inputs Y and Z and return a summary of the result. This is done with the following operations:

$$z' = g([z_0; z_1; ...; z_{l-1}])$$
$$z = \text{ATTENTION}(z', Y, Y),$$

where $g(\cdot) : \mathbb{R}^{(d_0 + d_1 + ... + d_{l-1})} \Rightarrow \mathbb{R}^{d_x}$ is a linear function, ; is the concatenation, and attention$(\cdot)$ is instantiated with the multi-head attention:

$$\text{ATTENTION}(z', Y, Y) = [\text{att}_1; \text{att}_2; ...; \text{att}_h]W^O$$

$$\text{att}_k = \text{att}(z'W_k^Q, YW_k^K, YW_k^V)$$

$$\text{att}(Q, K, V) = \text{SOFTMAX}(\frac{QK^{\text{T}}}{\sqrt{d_x}})V,$$

where W^O, W_k^Q, W_k^K and W_k^V are trainable parameters.

PREDICT The predict module, i.e. $S = \text{PREDICT}(Z, X)$, takes the list of output vectors $Z = [z_0, z_1, ..., z_{m-1}]$ from previous modules, where $z_k \in \mathbb{R}^{d_k}(0 \le k < m)$ and m is the number of previous modules, and the candidates $X = [x_0, x_1, ..., x_{n-1}]$, where $x_k \in \mathbb{R}_x^d$ and n is the number of candidates, and produces scores for the candidates. In our base model, each candidate is a token in the situation or question, and the score is a pair of numbers representing span start and end probabilities for that token. When we use this module in a re-ranker (Section 2.3), the candidates X are already encoded spans, and so we produce just one number for each span. The PREDICT module simply uses a linear scoring function on the concatenation of its inputs:

$$S = [s_0, s_1, ..., s_{n-1}]$$
$$s_k = \text{SCORE}([z_0; z_1; ...; z_l; x_k]),$$

where $\text{SCORE}(\cdot) : \mathbb{R}^{1 \times (d_0 + d_1 + ... + d_{m-1} + d_x)} \Rightarrow \mathbb{R}^r$ is a linear function, ; is the concatenation and $r = 2$ if the module is used to extract spans, while $r = 1$ if the module is used to score candidates for the reranker.

Full model Our full model combines these modules in the following way to compute span start and end scores for each token (depicted graphically in Figure 2):

$$S_{\text{ROBERTA}}, B_{\text{ROBERTA}}, Q_{\text{ROBERTA}} = P_{\text{ROBERTA}}$$
$$X = [S_{\text{ROBERTA}}; Q_{\text{ROBERTA}}]$$
$$z_0 = \text{SELECT}(Q_{\text{ROBERTA}})$$
$$z_1 = \text{SELECT}(B_{\text{ROBERTA}})$$
$$z_2 = \text{CHAIN}(B_{\text{ROBERTA}}, [z_0])$$
$$z_3 = \text{CHAIN}(S_{\text{ROBERTA}}, [z_1, z_2])$$
$$\bar{S}, \bar{E} = S = \text{PREDICT}([z_0, z_1, z_2, z_3], X)$$

2.3 Multi-Step Reranker

Most questions in ROPES have only two or three reasonable candidate answers (in Figure 1 these are "category A" and "category B"), and we find that the baseline model is able to reliably find these answers, though it has a hard time selecting between them. This suggests that a reranker that only focuses on deciding which of the candidates is correct could be effective. To do this, we take the top c spans output by the baseline system and score these

candidates directly using our MS-Inference model instead of producing span start and end scores for each input token.

Scoring spans instead of tokens To feed the candidate spans into our multi-step inference model, we represent each span as a single vector by concatenating its endpoint tokens: $x_{(i,j)} = [x_i; x_j]$. We take all c candidates and concatenate them together as X, instead of $X = [S; Q]$ as is done in our base model. Similarly, $\text{PREDICT}(Z, X)$ outputs a single score $\bar{O}$ per candidate instead of a pair of start and end probabilities.

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

3 Data bias in ROPES

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 we discovered that this led to a large distributional shift between train/dev and test, which we explore in this section. In light of this analysis, we recommend treating the dev set as an in-domain test set, and the original test set as an out-of-domain test.

Answer types Our analysis is based on looking at the syntactic category of the answer phrase. We use the syntactic parser of Kitaev and Klein (2018) to obtain constituent trees for the passages in ROPES. We take the constituent label of the lowest subtree that covers the answer span[3] as the answer type.

The four most frequent answer types in ROPES are noun phrase (NP), verb phrase (VP), adjective phrase (ADJP) and adverb phrase (ADVP). Table 1 shows examples for each type. Most NP answers

---

[3] The passages could have more than one span that matches the answer; we use the last occurrence of the answer span for our analysis.

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.

come from the situation, while the other answer types typically come from the question.

Bias The distribution of answer types in the train/dev/test sets of ROPES are shown in Table 2. We found that the distribution in the train set is similar to development set, where most of the answers are NPs (85%), with ADJP being the second most frequent. However, the test set has a very different distribution over answer types, where less than half of the answers NPs, and there are more VPs, ADJPs, ADVPs, and other types.

This distributional shift over answer types between train/dev and test raises 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 this section, we evaluate the performance of our proposed model relative to baselines on ROPES.

Answer type	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.

	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 distributional shift between dev and test (described in

3044

parameter	value
hidden size	1024
batch size	8
gradient accumulation	16
epoch	10
learning rate	1e-5
weight decay rate	0.1
warming up ratio	0.06
optimizer	adam
doc stride	150
maximum pieces	384

Table 4: The hyperparameters

Model	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 5: The exact match scores by three systems. For the first two columns, we performed model selection on dev; for the third column, we performed model selection on a separate train-dev set.

Section 3), 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 which is the same to the output dimension of ROBERTA-LARGE, and the number of heads on multi-step attentions is 8. All the models share the same hyperparameters that are shown in Table 4.[4]

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 5 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

---

[4]The hyperparamters are manually tuned according to the performance on dev dataset.

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, Table 5 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.

Ablation Study We perform an ablation study on the multi-step system and the multi-step reranker. Table 6 shows the results on dev set by various ablated systems. 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 (around) more than that of the multi-step system without B SELECT or S CHAIN (around -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 (around -5.9% EM) more than that of the multi-step reranker without B CHAIN (-3.7% EM).

3045

Model	EM
Multi-step	67.8
w/o Q SELECT	62.8 (-5.0)
w/o B CHAIN	62.3 (-5.5)
w/o B SELECT	65.9 (-1.9)
w/o S CHAIN	65.5 (-2.3)
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)

Table 6: 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.

Model	NP	VP	ADJP	ADVP	avg
Baseline	60.0	38.1	60.4	62.7	53.03
Multi-step	68.8	39.7	61.3	72.6	58.65
Multi-step reranker	71.8	38.1	63.8	75.0	60.52
+ensemble	75.0	42.9	61.3	78.6	62.75

Table 7: The exact match accuracy of most four frequent question types in test dataset. avg is the weighted accuracy in terms of frequency of the four kinds of questions.

Answer Types We break down the overall accuracy by answer type, which is shown in Table 7. 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 a 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, a well-trained model does not generate similar candidates for the training set to what it generates for the dev and test sets, due to overfitting to the training set. In order to get useful

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

Table 8: The average accuracy on training data for the multi-step reranker.

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 9: The oracle scores for top k candidates.

candidates for the training set, we need a model that was not trained on the data that it generates candidates for. We investigate four strategies based on cross-validation to generate training data candidates: 10-fold, 5-fold, 2-fold and 3-turn. With the k-fold method, the training data is partitioned into k parts, and $(k-1)$ parts are used to train a model that generates candidates answers for the remaining part. With the k-turn method, the training data is partitioned into k parts, and the ith part is used to train a model that generates candidate answers for $(i+1)$th part.

Table 8 shows the average accuracy on training data. The accuracy on training data generated 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. We adopt the 3-turn self-sampling method for our experiments.

Table 9 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 on the development set, giving a large jump over just two candidates, and this is what we used during our main experiments.

Error Analysis and Future Work   We analyze some errors that our proposed model made, aiming to discover the questions that our model could not cover. Table 10 shows some questions that our proposed model gives 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 comes from the difficulty of associating the relevant information with the correct entities. Future work can be motivated by the discovery to design more modules to deal with this phenomenon.

5   Related Work

Neural Module Networks   were originally proposed for visual question answering tasks (Andreas et al., 2016b,a), and recently have been used on several reading comprehension tasks (Jiang et al., 2019; Jiang and Bansal, 2019; Gupta et al., 2020), where they specialize the module functions such as FIND and COMPARE to retrieve the relevant entities with or without supervised signals for HotpotQA (Yang et al., 2018) or DROP (Dua et al., 2019). As ROPES is quite different from these datasets, the modules that we choose to use are also different, focusing on chained inference.

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; Feldman and El-Yaniv, 2019), QANGAROO (Welbl et al., 2018; Chen et al., 2019b; Zhuang and Wang, 2019; Tu et al., 2019) and WIKIHOP (Welbl et al., 2018; Song et al., 2018; Das et al., 2019; Asai 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

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
Ours: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
Ours: 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
Ours: Day 2

Table 10: The examples of the answers to the questions by the multi-step reranker.

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.

Acknowledgments

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.

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2019. Learning to retrieve reasoning paths over wikipedia graph for question answering. In *International Conference on Learning Representations*.

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.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. 2019. Multi-step retriever-reader interaction for scalable open-domain question answering. In *The International Conference on Learning Representations (ICLR)*.

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.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. 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 2368–2378.

Yair Feldman and Ran El-Yaniv. 2019. Multi-hop paragraph retrieval for open-domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2296–2309.

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.

Nitish Gupta, Kevin Lin, Dan Roth, Sameer Singh, and Matt Gardner. 2020. Neural module networks for reasoning over text. In *International Conference on Learning Representations*.

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.

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.

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

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 comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2252–2262, Florence, Italy. Association for Computational Linguistics.