An Empirical Analysis of Multiple-Turn Reasoning Strategies in Reading Comprehension Tasks

Yelong Shen[†], **Xiaodong Liu**[†], **Kevin Duh**[‡], **Jianfeng Gao**[†]
[†] Microsoft Research, Redmond, WA, USA
[‡] Johns Hopkins University, Baltimore, MD, USA
[†]{yeshen,xiaodl,jfgao}@microsoft.com [‡]kevinduh@cs.jhu.edu

Abstract

Reading comprehension (RC) is a challenging task that requires synthesis of information across sentences and multiple turns of reasoning. Using a state-of-the-art RC model, we empirically investigate the performance of single-turn and multiple-turn reasoning on the SQuAD and MS MARCO datasets. The RC model is an end-to-end neural network with iterative attention, and uses reinforcement learning to dynamically control the number of turns. We find that multiple-turn reasoning outperforms single-turn reasoning for all question and answer types; further, we observe that enabling a flexible number of turns generally improves upon a fixed multiple-turn strategy. We achieve results competitive to the state-of-the-art on these two datasets.

1 Introduction

There is an old Chinese proverb that says: *"Read a hundred times and the meaning will appear."* Several recent reading comprehension (RC) models have embraced this kind of multiple-turn strategy; they generate predictions by making multiple passes through the same text and integrating intermediate information in the process (Hill et al., 2016; Dhingra et al., 2016; Sordoni et al., 2016; Shen et al., 2016). While state-of-the-art results have been achieved by these models, there has yet to be an in-depth analysis of the impact of the multiple-turn strategy to reasoning. This paper attempts to fill this gap.

We provide empirical results and analysis on two challenging RC datasets: the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016), and the Microsoft Machine Reading Comprehension Dataset (MS MARCO) (Nguyen et al., 2016). Given a question Q, the RC model is to read passages P and produce an answer A, which could be free-form text or one of the possible candidate spans in the passage.

The following example from SQuAD illustrates the need for synthesis of information across sentences and multiple turns of reasoning:

Q: What collection does **the V&A Theator & Performance galleries** hold?

P: **The V&A Theator & Performance galleries** opened in March 2009. ... **They** hold the UK's biggest national collection of <u>material about live performance.</u>

To infer the answer (the underlined portion of the passage P), the model needs to first perform coreference resolution so that it knows "**They**" refers "**V&A Theator**", then extract the subspan in the direct object corresponding to the answer. This process can be modeled by the repeated processing of intermediate states and input in a neural net.

To perform the analysis, we adopt the ReasoNet model of Shen et al. (2016). This is an end-to-end neural network that uses an iterative attention mechanism to simulate multiple-turn reasoning in RC. It has achieved strong results on cloze-style RC tasks like CNN/DailyMail (Hermann et al., 2015) and we extend it to SQuAD and MS MARCO tasks. The advantage of using ReasoNet for our purpose is that it uses reinforcement learning to dynamically determine the number of turns for each question-passage pair. This enables us to analyze the behavior of multiple-turn reasoning in neural network models.

We find that multiple-turn reasoning outperforms single-turn reasoning across the board for various types of question and answer types. Furthermore, the flexibility to dynamically decide the

Proceedings of the The 8th International Joint Conference on Natural Language Processing, pages 957–966,
Taipei, Taiwan, November 27 – December 1, 2017 ©2017 AFNLP

	SQuAD	MS MARCO
query source	crowdsourced	user logs
answer (A)	span of words	free-form text
#questions (Q)	100K questions	100K queries
#passages (P)	23K paragraphs	1M paragraphs

Table 1: Dataset characteristics

number of turns generally improves over a fixed multiple-turn strategy, where the number of turns are set a priori. As an additional contribution, our extension to the ReasoNet model achieves results competitive with the state-of-the-art on SQuAD and MS MARCO.

In the following, Section 2 describes our two RC tasks, Section 3 explains the model we used for analysis, and Section 4 discusses results.

2 Reading Comprehension Tasks

We focus this study on two RC tasks which we believe require sophisticated reasoning.

SQuAD: SQuAD is a machine comprehension dataset constructed on 536 Wikipedia articles (23K paragraphs), with more than 100,000 questions. In contrast to prior datasets such as (Richardson et al., 2013; Hermann et al., 2015), SQuAD does not provide a multiple choice list of answer candidates. Instead, the RC model must select the answer from all possible spans in the passage. Crowdsourced workers are asked to read each passage (a paragraph), come up with questions, and then mark the answer spans.

There is a variety of questions and answers. The authors of SQuAD described several types of reasoning required to answer questions: (a) lexical variation between question (Q) and answer (A) that can be solved by understanding synonyms, (b) lexical variation that could be solved by world knowledge, (c) syntactic variation between Q and A sentence, and (d) multiple sentence reasoning that require anaphora or higher-level fusion.

The 100K (question, passage, answer) tuples is randomly partitioned into a training (80%), a development (10%) and test set splits (10%). Two evaluation metrics are used: Exact Match (EM), which measures the percentage of span predictions that matched any one of the ground truth answer exactly, and Macro-averaged F1 score, which measures the average overlap between the prediction and the ground truth answer. Human performance on the test set is 82.3% EM and 91.2% F1.

MS MARCO: MS MARCO is a large scale real-world RC dataset that contains 100,000 queries collected from anonymized user logs from the Bing search engine. The characteristic of MS MARCO is that all the questions are real user queries and passages are extracted from real web documents. The data is constructed as follows: for each question/query Q, up to approximately 10 passages P are extracted from public web documents and presented to human judges. These passages might potentially have the answer to the question, and are selected through a separate information retrieval system. The judges write down answers in free-form text, and according to the authors of MS MARCO, the complexity of answer varies from a single "yes/no" or entity name (e.g. Q: "What is the capital of Italy"; A: Rome), to long textual answers (e.g. Q: "What is the agenda for Hollande's state visit to Washington?"). Long textual answers may need to be derived through reasoning across multiple pieces of text.

The dataset is partitioned into a 82,430 training, a 10,047 development, and 9,650 test tuples. Since the answer is free-form text, the evaluation metrics of choice are BLEU (Papineni et al., 2002) and ROUGE-L (Lin, 2004). To apply the same RC model to both SQuAD (where answers are text spans in P) and MS MARCO (where answer are free-form text), we search for spans in MS MARCO's passages that maximizes the ROUGE-L score with the raw free-form answers. Our training data uses these spans as labels, but we evaluate our model with respect to the raw free-form answers; this has an upper bound of 94.23 BLEU and 87.53 ROUGE-L on the dev set. By this construction, there are multiple number of passages to read for each question, but the answer span might only involve a few passages (i.e. the ones that include the max ROUGE substring). We describe techniques to handle this case in Section 3.2.

3 Model: ReasoNet++

The reading comprehension task involves a question/query $Q = \{q_0, q_1, ..., q_{m-1}\}$ and a passage $P = \{p_0, p_1, p_{n-1}\}$ and aims to find an answer span $A = \langle a_{start}, a_{end} \rangle$ in P. Here, m and n denote the number of tokens in Q and P, respectively, while a_{start} and a_{end} indicate the indices of tokens in P. The learning process for reading comprehension is to learn a function $f(Q, P) \rightarrow A$ trained on a set of tuples $\langle Q, P, A \rangle$.

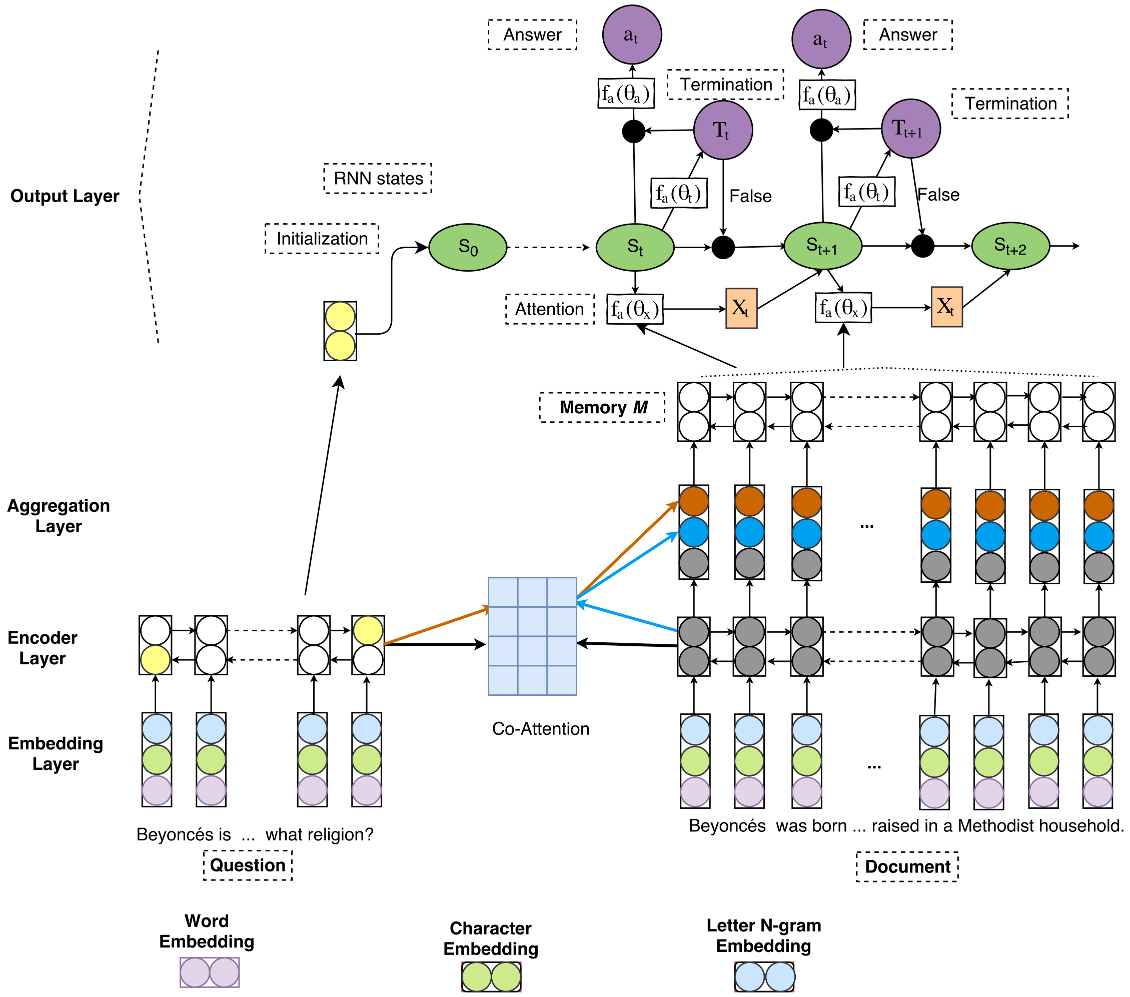


Figure 1: **Architecture of ReasoNet++**: The embedding/encoder layers compute representations for the question Q and the passage document P. The aggregation layer uses co-attention to compute question-aware passage information and passage-aware question information. Then a GRU combines these information into memory cells and feeds them to the output layer. The output layer models the multiple-turn reasoning mechanism, where intermediate results are stored in S_t and the answer is generated only when the termination signal is triggered. Each S_t is a recurrent network state and models one turn of reasoning.

Our model **ReasoNet++**, is an extension of ReasoNet (Shen et al., 2016) with three enhancements: (1) In the input layer, we added character and letter 3-gram embeddings to improve robustness to rare words. (2) We implemented co-attention (Seo et al., 2016) in the aggregation layer to focus on relevant words in both Q and P. (3) For the MS MARCO task, which needs to handle multiple passages, we incorporated an extra passage ranker component. The architecture is shown in Figure 1. In brief, the embedding/encoder layers first build representations of Q and P. The aggregation layer uses co-attention to fuse information from the Q-P pair. The output layer is a recurrent net that maintains intermediate state and dynamically decides at which turn to generate the answer.

3.1 Detailed description of ReasoNet++

Embedding Layer: We adopt three types of embeddings to represent input word tokens in Q and P. For word embeddings, we use pre-trained GloVe vectors (Pennington et al., 2014). To address the out-of-vocabulary problem, we also include character and letter 3-gram embeddings. Character embeddings are fed into a convolutional neural network (CNN) as in (Kim, 2014), then max-pooled to form a fixed-size vector for each token. For letter 3-gram embeddings, we follow Huang et al. (2013) by first *hashing* each word as a bag of letter 3-gram, then feeding them into another CNN. The concatenation of all embeddings are then fed to a two-layer Highway Network (Srivastava et al., 2015). Therefore, we obtain the final embedding for the words in Q as a matrix $E^q \in \mathbb{R}^{d\times m}$, and words in P as $E^p \in \mathbb{R}^{d\times n}$, where d is the dimension of the embedding.

Encoding Layer: On the top of embedding layer, we utilize a bidirectional Gated Recurrent Unit (GRU), a variety of the Long Short-Term Memory Network (LSTM) (Hochreiter and Schmidhuber, 1997), to encode the words in context. We obtain $H^q \in \mathbb{R}^{2d\times m}$ as the representation of Q and $H^p \in \mathbb{R}^{2d\times n}$ as the representation of P.

Aggregation Layer: In this layer, we construct the memory, a summary of information from both the Q and P, for each word in P. A co-attention mechanism (Seo et al., 2016), which attends to Q and P simultaneously, is applied by first computing an alignment matrix in two directions: from Q to P and from P to Q. The alignment matrix C measures the similarity between Q and P:

$$C = f_{match}(H^q, H^p) \in \mathbb{R}^{m\times n} \qquad (1)$$

The element at i-th row and j-th of the alignment matrix, C_{ij}, indicates the similarity between i-th word in the question and j-th word in the passage. In detail, $C_{ij} = f_{match}(H^q_{:i}, H^p_{:j})$ is a trainable scalar function that measures the similarity between two input vectors, $H^q_{:i}$, which is the i-th column vector of H^q, and $H^p_{:j}$, which is the j-th column vector of H^p. We parameterize $f_{match}(a, b) = w_C^T[a; b; a \circ b]$, where $\circ$ denotes the Hadmard product, $[;]$ indicates vector concatenation across rows, and $w_C \in \mathbb{R}^{6d}$ is a trainable weight vector. We normalize C row-wise to produce the attention weight across the passage for each word of the question:

$$C^q = softmax(C) \in \mathbb{R}^{m\times n}. \qquad (2)$$

To measure which context words in the P have the closest similarity to the words in the Q, we define an attention weight on the words in passage as:

$$c^p = softmax(max_{col}(C))^T \in \mathbb{R}^n. \qquad (3)$$

The final context representation of the P is:

$$U = f_{agg}(H^p, H^qC^q, \sum_{i=0}^{i=m-1} H^p_{:i}c^p) \in \mathbb{R}^{8d\times n}. \qquad (4)$$

In our experiment, we define $f_{agg}(B, C, D) = [B; C; B \circ C; B \circ D]$. Note that B, C, D are matrices with the same dimension, $\circ$ denotes the Hadmard product and ; indicates matrix concatenation across columns. Note that since $H^p, H^qC^q, \sum_{i=0}^{i=m-1} H^p_{:i}c^p$ are all $2d$ by n matrices, U is a $8d$ by n matrix. Finally, to incorporate the full context, the "memory cells" of the passage are computed by a bidirectional GRU on top of U:

$$M^p = BiGRU(U) \in \mathbb{R}^{2d\times n} \qquad (5)$$

Output Layer: This layer dynamically decides when to stop reasoning and output the answer. A recurrent neural network (Rumelhart et al., 1986; Elman, 1990) is adopted to maintain the states of the reasoning process. Formally, the t-th time step of inference state is denoted as S_t, and the next state is defined by $S_{t+1} = GRU(S_t, X_t)$. Note that the X_t is an attention vector generated based on the current state and the memory of the passage: $X_t = f_a(S_t, M^p)$ as in (Shen et al., 2016). Specifically, the attention score $a_{t,i}$ on a memory vector $m_i \in M^p$ given a state S_t is computed as $a_{t,i} = softmax_{i=1,\ldots,|M^p|}\lambda\text{cosine}(w_1m_i, w_2S_t)$, where λ is set to 10 in our experiments and the projection matrices w_1 and w_2 map the memory vector and state into the same space, they are learned during training. The attention vector X_t can be written as $X_t = \sum_i^{|M|} a_{t,i}m_i$. The initial state S_0 of the inference is from the encoding representation of the question (we pick the last state of the forward GRU and the backward GRU in the H^q).

The *termination gate* will produce a stochastic random variable according to the current inference state: $T_t \sim p(\cdot|f_t(S_t))$, where f_t is modeled by a $2d\times 10\times 10\times 10\times 1$ feed-forward neural network. Note T_t is a binary random variable: if T_t is true, the recurrent net will stop and the answer model will execute; otherwise it will generate an attention vector X_{t+1} and update the next state S_{t+1}.

The answer module needs to output a span in passage. We do this with two feedforward networks, one predicting the start point of the span and the other predicting the end point, so predicted answer at turn t is $a_t = (y_s^t, y_e^t)$:

$$y_s^t = softmax(w_s^T[M^p, (w_{ps}^T M^p) \circ S_t]) \qquad (6)$$

$$y_e^t = softmax(w_e^T[M^p, (w_{pe}^T M^p) \circ S_t]). \qquad (7)$$

where w_s, w_e, w_{ps} and w_{pe} are trainable model parameters. Since the termination state is discrete and is not connected to the final output directly, we use the Contrastive Reward method (Shen et al., 2016) inspired by deep reinforcement learning (Weissenborn, 2016; Mnih et al., 2014) for training.

3.2 Passage ranking extension

The MS MARCO dataset provides multiple passages per question/query. Our architecture in Figure 1 is built for a single passage-question pair, so we need to extend it to handle multiple passages. We propose a solution using passage ranking. Assume there are J passages, $P^{(1)}, \ldots, P^{(J)}$. First, our model runs independently on every $(P^{(j)}, Q)_{j=1,\ldots,J}$ pair, generating J different answer spans (empty spans are possible). Then, we multiply the probability of each answer span with a score $r(P^{(j)}, Q)$ provided by a passage ranker, and output the answer with the maximum combined score, similar to EpiReader (Trischler et al., 2016). The passage ranker is a information retrieval model (Shen et al., 2014).[1] It can be trained on the same RC data, where documents with answers are considered relevant and those that do not are considered irrelevant.

All our MS MARCO results use the passage ranking extension, unless otherwise mentioned.

4 Experiments

We seek to answer the following questions:

1. Is multiple-turn reasoning beneficial for RC? (Section 4.1)

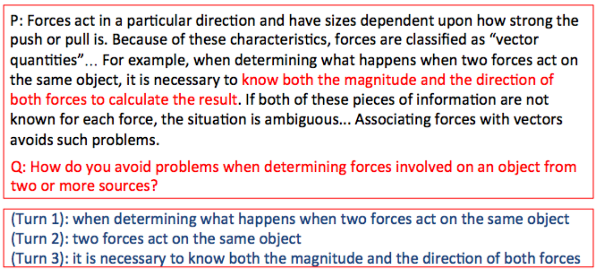
2. What types of questions/answers benefit most from multiple-turn reasoning? (Section 4.2)

3. How many turns are employed in practice by ReasoNet++, and what are the implications for dynamic versus fixed strategies in multiple-turn reasoning? (Section 4.3)

In addition to the above analyses, we also demonstrate that our ReasoNet++ achieves state-of-the-art results (Section 4.4) and discuss some ablation studies on model variants (Section 4.5).[2]

	SQuAD	MS MARCO
Single model	EM/F1 Score	BLEU/ROUGE-L
Single turn	67.8/76.7	33.65/36.54
Fixed 5-turn	70.1/78.9	34.93/36.67
ReasoNet++	**70.8/79.4**	**38.62/38.01**

Table 2: **Main results**—Comparison of single turn to multiple turn reasoning strategies on SQuAD and MS MARCO dev sets. Both multiple turn strategies (fixed at 5, or dynamically decided based on ReasoNet++) outperform Single turn in all metrics. The dynamic strategy further improves upon the fixed multiple 5-turn strategy.



P: Forces act in a particular direction and have sizes dependent upon how strong the push or pull is. Because of these characteristics, forces are classified as "vector quantities"... For example, when determining what happens when two forces act on the same object, it is necessary to know both the magnitude and the direction of both forces to calculate the result. If both of these pieces of information are not known for each force, the situation is ambiguous... Associating forces with vectors avoids such problems.

Q: How do you avoid problems when determining forces involved on an object from two or more sources?

(Turn 1): when determining what happens when two forces act on the same object
(Turn 2): two forces act on the same object
(Turn 3): it is necessary to know both the magnitude and the direction of both forces

Figure 2: Case study from SQuAD of answers from multiple turns. In Turn 1, the model identifies a span similar to the question. This is refined and at Turn 3 a better answer becomes attainable.

4.1 Is multiple-turn reasoning beneficial?

In summary, yes. We compare three systems:

Single turn: the RC model only has one turn of reasoning. This corresponds to a model like Figure 1 without termination nodes, where the output layer always stops at $S_{t=1}$.

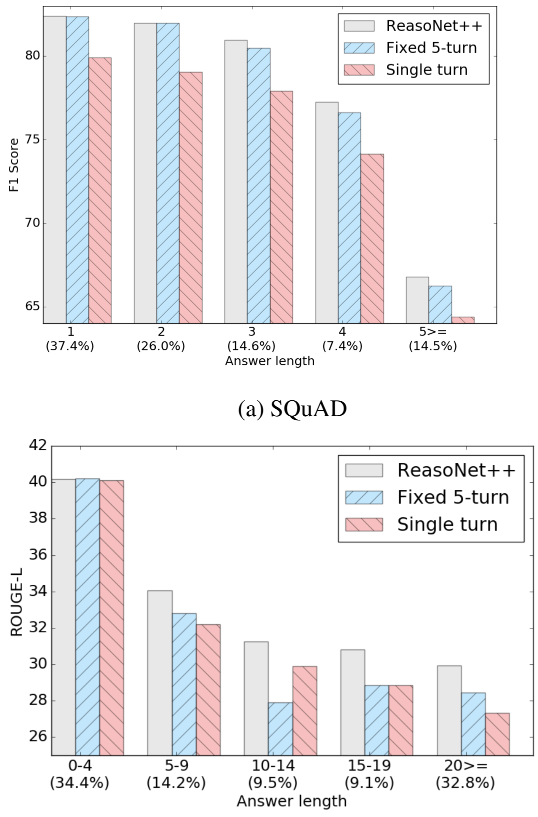
Fixed 5-turn: the RC model runs 5 turns of iterative attention. This is Figure 1 without termination nodes, where output layer always stops at $S_{t=5}$.

ReasoNet++ (Dynamic multiple-turn reasoning): this is the RC model in Figure 1, which can decide from 1 to T turns based on the termination probability on each Q-P pair at test time. We set $T = 5$ to compare with the Fixed 5-turn system.

The main results are shown in Table 2. We observe that both multiple turn strategies (either fixed at 5 turns, or dynamically decided based on ReasoNet++) outperform the single turn system in all metrics. The dynamic strategy further improves upon the fixed multiple 5-turn strategy. For ex-

[1] Our implementation first hashes words into letter 3-gram (50K dimension), then use a CNN with 256 hidden nodes and the size of window 5, and lastly optimizes the similarity between the vector representations of P and Q.

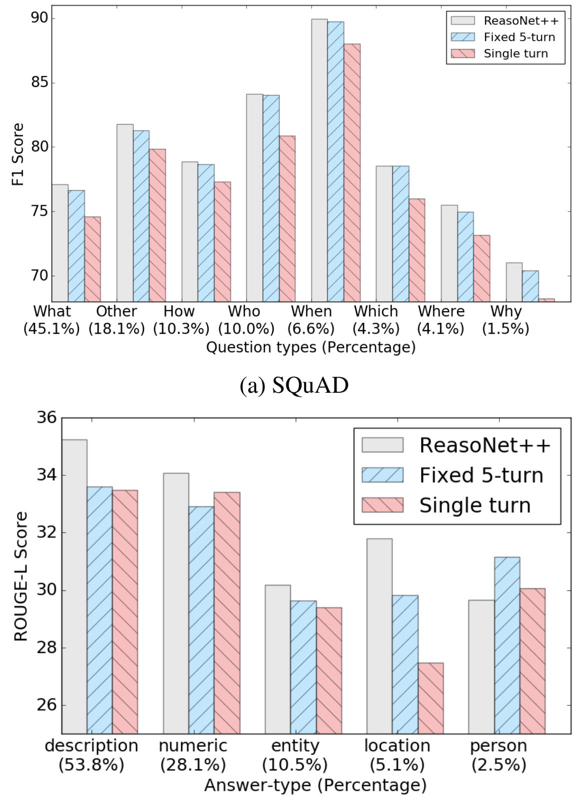
[2] A note on hyperparameters: Throughout all experiments, we use NLTK to tokenize P and Q, and employ pre-trained case-sensitive 300 dimension GloVe embeddings[3]. A one layer CNN with 100 dimensions and window size of 5 is used to compute the character embeddings; a one layer CNN with 100 dimension and window size of 1 is used for letter 3-gram embeddings. The size of hidden nodes of all GRU's is set to 128. A five layers feedforward network $(2d(256) \times 10 \times 10 \times 10 \times 1)$ is used for the terminate network and the maximum number of reasoning turns in the recurrent net is capped at 5. To avoid overfitting, we adopt 0.15 dropout rate over the letter 3-gram and character embeddings, and 0.25 dropout rate (Srivastava et al., 2014) over GRU network. The model is optimized with AdaDelta (Zeiler, 2012) with an initial learning rate 0.5.



(a) SQuAD

(b) MS MARCO

Figure 3: Score breakdown by answer length



(a) SQuAD

(b) MS MARCO

Figure 4: Score breakdown by query/answer type

ample, the F1 score on SQuAD improves from 76.7 to 78.9 when increasing the number of turns from 1 to 5, and further improves to 79.4 with dynamic multiple turns. On MS MARCO, we see a ROUGE improvement from 36.54 (1-turn) to 36.67(5-turn) and 38.62 (dynamic multi-turn). These results convincingly show that multiple-turn reasoning is helpful for SQuAD and MS MARCO tasks. Figure 2 shows a case study of how answers improve with each turn.

4.2 What types of questions/answers benefit most from multiple-turn reasoning?

We find that improvements from multiple-turn reasoning is generally seen across the board, but particularly helps questions with longer answers. Figure 3 shows the score breakdown of Table 2 according to answer length (# of words). For SQuAD, both ReasoNet++ and Fixed 5-turn outperform Single turn for all answer lengths, and ReasoNet++ outperforms Fixed 5-turn for answer lengths > 3. For MS MARCO, ReasoNet++ outperforms Fixed 5-turn for answer lengths > 5; on the other hand, there is almost no difference among systems for short answers (0-4). We hypothesize there is a correlation between answer length and the difficulty of the question; for difficult questions there may be more potential for multiple-turn reasoning to improve results.

We also visualize the score breakdown according to question/answer type (Figure 4). For MS MARCO, the questions are annotated by the type of the correct answer: description (e.g. Q: “How to cook a turkey”), numeric (e.g. Q: “Xbox one release data”), entity, location, person. There is no such annotation for the entire SQuAD dev data, but we can classify questions by their first word: What, Who, When, Which, etc. Similar to the answer length results, we observe that multiple-turn reasoning outperforms single turn for SQuAD across the board, regardless of question type. For MS MARCO, ReasoNet++ gave large improvements over single turn in particular for description and location types. Descriptions tend to be lengthy, so this again corroborates our hypothesis that there may be more potential gains for questions requiring long answers.

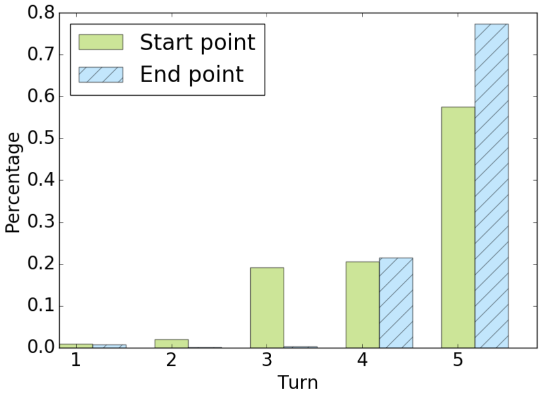


Figure 5: Distribution on the number of turns by ReasoNet++on the SQuAD dev set. Note that start points are often decided before end points, and most answer spans are generated after 3 turns.

4.3 How many turns of reasoning are employed in practice?

We are interested in understanding the number of turns determined by ReasoNet++. When does it decide to terminate? In Figure 5, we plot the distribution of turns until the model decides on start points and end points (of the answer span).

First, note the start point is often decided before the end point, e.g. the start is already determined at turn 3 for approximately 20% of the questions , but the end does not get predicted until turn 4 or 5. Intuitively, we think it is easier to first identify the start of an answer, then use that signal as intermediate state S_t to identify the end point.

Second, there is almost no termination at turns 1 or 2, implying the model prefers more iterations of reasoning. Most terminations are done at step 4 or 5, which explains the relatively close performance results between Fixed 5-turn and ReasoNet++.

4.4 Comparison with state-of-the-art

Our ReasoNet++ model, which is an extension of ReasoNet (Shen et al., 2016), achieves scores competitive with state-of-the-art results. The official leaderboard results are shown in Table 3 (MS MARCO) and Table 4 (SQuAD) Results are divided by whether we use an individual model or an ensemble of models. For SQuAD, the ReasoNet++ ensemble model achieves the best EM and F1 test score among all published works, and places second if we include r-net. Similarly, the ReasoNet++ individual model results are in the top 1 or 2 ranks, competitive with published works like Zhang et al. (2017) and Weissenborn et al. (2017). For MS MARCO (Table 3), ReasoNet++ ranks first in test ROUGE and second in test BLEU (after Match-LSTM (Wang and Jiang, 2016)). Note that some of the models on the leaderboard use multiple-turn reasoning, while others do not. But we refrain from drawing conclusions about multiple-turn reasoning by comparing across models, due to other confounding variables, e.g. different embeddings and network architectures.

System	BLEU/ROUGE-L	
	Dev Set	Test Set
ReasoNet++ ^{Individual}	38.62/38.01	39.86/38.81
Match-LSTM	-/-	40.72/37.33
FastQA_Ext	35.0/34.4	33.93/33.67
FastQA	34.9/33.0	33.99/32.09
Human Performance	-/-	46/47

Table 3: Official MS MARCO leaderboard performance on April 5, 2017.

4.5 Ablation studies and model variants

We now present some ablation studies to demonstrate the differences between our ReasoNet++ and the original ReasoNet (Shen et al., 2016) in which we are based on.[4]

First, Table 5 shows the improvement from adding sub-word level modeling to ReasoNet, which only used word embeddings. We observe marked improvements of update number +1.1 F1 in SQuAD and +0.9 ROUGE in MS MARCO. Although these improvements are not as large as those we achieved with multiple-turn reasoning, they are are still considerable and imply that robust representations of words is an important building block to strong RC models.

Secondly, Table 6 shows the impact of passage ranking—this is only relevant for MS MARCO, which contains multiple passages for each question/query. Recall that the RC model needs to read approximately 10 passages to answer each query, and on average only one or two passage contain answer spans. ReasoNet++ extracts answer spans from each passage independently, then combines with an IR model to output the final answer. If we assume oracle ranking from the IR model, we can achieve 62 BLEU / 63 ROUGE, suggesting that

[4] Due to time constraints, we only perform ablation studies on the embedding and passage ranking enhancements, and leave the study of the impact of co-attention to future work.

Ensemble model results:	Dev Set (EM/F1)	Test Set (EM/F1)
r-net*	-/-	76.9/84.0
ReasoNet++ (Ensemble model)	75.4/82.9	75.0/82.6
BiDAF (Seo et al., 2016)	73.3/81.1	73.7/81.5
Multi-Perspective Matching (Wang et al., 2016)	69.4/78.6	73.8/81.3
Dynamic Coattention Networks (Xiong et al., 2016)	70.3/79.4	71.6/80.4
Match-LSTM with Ans-Ptr (Wang and Jiang, 2016)	67.6/76.8	67.9/77.0
Fine-Grained Gating(Yang et al., 2017)	62.4/73.4	62.4/73.3
Individual model results:		
r-net*	-/-	72.3/80.7
jNet (Zhang et al., 2017)	-/-	70.6/79.8
Ruminate Reader*	-/-	70.6/79.5
ReasoNet++ (Individual model)	70.8/79.4	70.6/79.36
Document Reader*	-/-	70.7/79.35
FastQAExt (Weissenborn et al., 2017)	70.3/78.5	70.8/78.9
RaSoR (Lee et al., 2016)	66.4/74.9	70.0/77.7
BiDAF (Seo et al., 2016)	67.7/77.3	68.0/77.3
Iterative Co-attention Network*	-/-	67.5/76.8
Dynamic Coattention Networks (Xiong et al., 2016)	65.4/75.6	66.2/75.9
Match-LSTM with Bi-Ans-Ptr (Wang and Jiang, 2016)	64.1/73.9	64.7/73.7
Attentive CNN context with LSTM*	-/-	63.3/73.5
Dynamic Chunk Reader (Wang and Jiang, 2016)	62.5/71.2	62.5/71.0
LR baseline (Rajpurkar et al., 2016)	40.0/51.0	40.4/51.0
Human Performance	80.3/90.5	82.3/91.2

Table 4: Official SQuAD leaderboard performance on April 5, 2017. Asterisk * denotes unpublished works. Results are sorted by Test F1.

System	SQuAD	MS MARCO
	EM/F1 Score	BLEU/ROUGE
word+char+3gram	70.8/79.4	38.62/38.01
word+char	70.4/79.1	38.37/37.91
word	69.9/78.3	37.77/37.14

Table 5: Comparison of input embeddings: the addition of character (char) and letter trigram (3gram) embeddings to word embeddings (word) clearly improve results on SQuAD and MS MARCO development sets.

	BLEU/ROUGE-L
Oracle passage selection	62.83/63.17
Passage ranking	38.62/38.01

Table 6: Effect of multiple passages per query in MS MARCO.

better passage ranking models (e.g. via joint training with RC models) is fruitful as future work.

5 Conclusion

This paper empirically investigates the performance of single-turn and multiple-turn reasoning on two challenging reading comprehension tasks: SQuAD and MS MARCO. To perform the analysis, we adopt the neural network model of Shen et al. (2016), which employs iterative attention and uses reinforcement learning to dynamically control the number of turns. We find that multiple-turn reasoning outperforms single-turn reasoning for all question and answer types; further, we observe that enabling a flexible number of turns generally improves upon a fixed multiple-turn strategy. While our analysis is based on a single model, we believe the conclusions will be valuable for most RC methods using attention-based neural network. Our model extension to (Shen et al., 2016) achieves results competitive to the state-of-the-art on both tasks. As future work, we plan to investigate the impact of even deeper layers of reasoning and explore fast training methods to make such methods practical for large-scale datasets.

References

Bhuwan Dhingra, Hanxiao Liu, William W Cohen, and Ruslan Salakhutdinov. 2016. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549* .

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science* 14(2):179–211.

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

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. *ICLR* .

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

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM, pages 2333–2338.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1746–1751. http://www.aclweb.org/anthology/D14-1181.

Kenton Lee, Tom Kwiatkowski, Ankur Parikh, and Dipanjan Das. 2016. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436* .

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries.

Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. 2014. Recurrent models of visual attention. In *Advances in neural information processing systems*. pages 2204–2212.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 311–318.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text pages 2383–2392. https://aclweb.org/anthology/D16-1264.

Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 193–203.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *Cognitive modeling* 5(3):1.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603* .

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, pages 101–110.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2016. Reasonet: Learning to stop reading in machine comprehension. *arXiv preprint arXiv:1609.05284* .

Alessandro Sordoni, Philip Bachman, Adam Trischler, and Yoshua Bengio. 2016. Iterative alternating neural attention for machine reading. *arXiv preprint arXiv:1606.02245* .

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387* .

Adam Trischler, Zheng Ye, Xingdi Yuan, Philip Bachman, Alessandro Sordoni, and Kaheer Suleman. 2016. Natural language comprehension with the epireader. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 128–137. https://aclweb.org/anthology/D16-1013.

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905* .

Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. 2016. Multi-perspective context matching for machine comprehension. *arXiv preprint arXiv:1612.04211* .

Dirk Weissenborn. 2016. Separating answers from queries for neural reading comprehension. *arXiv preprint arXiv:1607.03316* .

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Fastqa: A simple and efficient neural architecture for question answering. *arXiv preprint arXiv:1703.04816* .

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604* .

Fan Yang, Zhilin Yang, and William W Cohen. 2017. Differentiable learning of logical rules for knowledge base completion. *arXiv preprint arXiv:1702.08367* .

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Junbei Zhang, Xiaodan Zhu, Qian Chen, Lirong Dai, and Hui Jiang. 2017. Exploring question understanding and adaptation in neural-network-based question answering. *arXiv preprint arXiv:1703.04617* .