R^3: Reinforced Ranker-Reader for Open-Domain Question Answering

Shuohang Wang[1*], Mo Yu[2], Xiaoxiao Guo[2], Zhiguo Wang[2], Tim Klinger[2], Wei Zhang[2]
Shiyu Chang[2], Gerald Tesauro[2], Bowen Zhou[3], and Jing Jiang[1]
[1]School of Information System, Singapore Management University
[2]AI Foundations - Learning, IBM Research AI. Yorktown Heights NY, USA
[3]JD.COM. Beijing, China
shwang.2014@smu.edu.sg, yum@us.ibm.com, xiaoxiao.guo@ibm.com

Abstract

In recent years researchers have achieved considerable success applying neural network methods to question answering (QA). These approaches have achieved state of the art results in simplified closed-domain settings[1] such as the SQuAD (Rajpurkar et al. 2016) dataset, which provides a pre-selected passage, from which the answer to a given question may be extracted. More recently, researchers have begun to tackle *open-domain QA*, in which the model is given a question and access to a large corpus (e.g., wikipedia) instead of a pre-selected passage (Chen et al. 2017a). This setting is more complex as it requires large-scale search for relevant passages by an information retrieval component, combined with a reading comprehension model that "reads" the passages to generate an answer to the question. Performance in this setting lags well behind closed-domain performance.

In this paper, we present a novel open-domain QA system called *Reinforced Ranker-Reader* (R^3), based on two algorithmic innovations. First, we propose a new pipeline for open-domain QA with a *Ranker* component, which learns to rank retrieved passages in terms of likelihood of extracting the ground-truth answer to a given question. Second, we propose a novel method that jointly trains the *Ranker* along with an answer-extraction *Reader* model, based on reinforcement learning. We report extensive experimental results showing that our method significantly improves on the state of the art for multiple open-domain QA datasets. [2]

Introduction

Open-domain question answering (QA) is a key challenge in natural language processing. A successful open-domain QA system must be able to effectively retrieve and comprehend one or more knowledge sources to infer a correct answer. Knowledge sources can be knowledge bases (Berant et al. 2013; Yu et al. 2017) or structured or unstructured text passages (Ferrucci et al. 2010; Baudiš and Šedivỳ 2015).

Q:	What is the largest island in the Philippines?
A:	**Luzon**
P1	Mindanao is the second largest and easternmost island in the Philippines.
P2	As an island, **Luzon** is the Philippine's largest at 104,688 square kilometers, and is also the world's 17th largest island.
P3	Manila, located on east central **Luzon** Island, is the national capital and largest city.

Table 1: An open-domain QA training example. Q: question, A: answer, P: passages retrieved by an IR model and ordered by IR score.

Recent deep learning-based research has focused on open-domain QA based on large text corpora such as Wikipedia, applying information retrieval (IR) to select passages and reading comprehension (RC) to extract answer phrases (Chen et al. 2017a; Dhingra, Mazaitis, and Cohen 2017). These methods, which we call *Search-and-Reading QA* (SR-QA), are simple yet powerful for open-domain QA. Dividing the pipeline into IR and RC stages leverages an enormous body of research in both IR and RC, including recent successes in RC via neural network techniques (Wang and Jiang 2017b; Wang et al. 2016; Xiong, Zhong, and Socher 2017; Wang et al. 2017).

The main difference between training SR-QA and standard RC models is in the passages used for training. In standard RC model training, passages are manually selected to guarantee that ground-truth answers are contained and annotated within the passage (Rajpurkar et al. 2016).[3] By contrast, in SR-QA approaches (Chen et al. 2017a; Dhingra, Mazaitis, and Cohen 2017), the model is given only QA-pairs and uses an IR component to retrieve passages similar to the question from a large corpus. Depending on the quality of the IR component, retrieved passages may not contain or entail the correct answer, making RC training more difficult. Table 1 shows an example which illustrates the difficulty. This ordering was produced by an off-the-shelf IR engine using the BM25 algorithm. The correct answer is contained in passage P2. The top passage (P1), despite being ranked highest by the IR engine, is ineffective for

[*] Word done when the author was at IBM.

[1]In the QA community, "openness" can be interpreted as referring either to the scope of question topics or to the breadth and generality of the knowledge source used to answer each question. Following (Chen et al. 2017a) we adopt the latter definition.

[2]Code: https://github.com/shuohangwang/mprc.

[3]This forms a closed-domain QA by our adopted definition where the domain consists of the given passage only.

answering the question, since it fails to capture the semantic distinction between "largest" and "second largest". Passage P3 contains the answer text ("Luzon") but does not semantically entail the correct answer ("Luzon is the largest island in the Philippines"). Training on passages such as P1 and P3 can degrade performance of the RC component.[4]

In this paper we propose a new approach which explicitly separates the tasks of predicting the likelihood that a passage provides the answer, and reading those passages to extract correct answers. Specifically we propose an end-to-end framework consisting of two components: a *Ranker* and a *Reader* (i.e. RC model). The Ranker selects the passage most likely to entail the answer and passes it to the Reader, which reads and extracts from that passage. The Reader is trained using SGD/backprop to maximize the likelihood of the span containing the correct answer (if one exists). The Ranker is trained using REINFORCE (Williams 1992) with a reward determined by how well the Reader extracts answers from the top-ranked passages. This optimizes the Ranker with an objective determined by end-performance on answer prediction, which provides a strong signal to distinguish passages lexically similar to but semantically different from the question.

We discuss the Ranker-Reader model in detail below but briefly, the Ranker and Reader are implemented as variants of Match-LSTM models (Wang and Jiang 2016). These models were originally designed for solving the text entailment problem. For this task, different non-linear layers are added for selecting the passages or predicting the start and end positions of the answer in the passage.

We evaluate our model on five different datasets and achieve state-of-the-art results on four of the them. Our results also show the merits of employing a separate REINFORCE-trained ranking component over several challenging fully supervised baselines.

Framework

Problem Definition We assume that we have available a factoid question $\mathbf{q}$ to be answered and a set of passages which may contain the ground-truth answer $\mathbf{a}^g$. Those passages[5] are the top N retrieved from a corpus by an IR model supplied with the question, for N a hyper-parameter. During training we are given only the $(\mathbf{q}, \mathbf{a}^g)$ pairs, together with an IR model with index built on an open-domain corpus.

Framework Overview An overview of the Ranker-Reader model is shown in Figure 1. It shows two key components: a **Ranker**, which selects passages from which an answer can be extracted, and a **Reader** which extracts answers from supplied passages. Both the Ranker and Reader compare the question to each of the passages to generate passage representations based on how well they match the question. The Ranker uses these "matched" representations to select a single passage which is most likely to contain the answer. The selected passage is then processed by the Reader to extract an answer sequence. We train the reader using SGD/backprop and produce a reward to train the Ranker via REINFORCE.

R^3: Reinforced Ranker-Reader

In this section, we first review the Match-LSTM (Wang and Jiang 2016) which provides input for both the Reader and Ranker. We then detail the Reader and Ranker components, and the procedure for joint training, including the objective function used for RL training.

Passage Representation Using Match-LSTM To effectively rank and read passages they must be matched to the question. This comparison is performed with a Match-LSTM, a state-of-the-art model for text entailment, shown on the right in Figure 1. Match-LSTMs use an attention mechanism to compute word similarities between the passage and question sequences. These are first encoded as matrices $\mathbf{Q}$ and $\mathbf{P}$, respectively, by a Bidirectional LSTM (BiLSTM) with hidden dimension l. With Q words in question $\mathbf{Q}$ and P words in passage $\mathbf{P}$ we can write:

$$\mathbf{H}^{\mathrm{p}} = \mathrm{BiLSTM}(\mathbf{P}), \quad \mathbf{H}^{\mathrm{q}} = \mathrm{BiLSTM}(\mathbf{Q}), \tag{1}$$

where $\mathbf{H}^{\mathrm{p}} \in \mathbb{R}^{l \times P}$ and $\mathbf{H}^{\mathrm{q}} \in \mathbb{R}^{l \times Q}$ are the hidden states for the passage and the question. In order to improve computational efficiency without degrading performance, we simplify the attention mechanism of the original Match-LSTM by computing the attention weights $\mathbf{G}$ as follows:

$$\mathbf{G} \quad = \quad \mathrm{SoftMax}\left((\mathbf{W}^{\mathrm{g}}\mathbf{H}^{\mathrm{q}} + \mathbf{b}^{\mathrm{g}} \otimes \mathbf{e}_Q)^{\mathrm{T}}\mathbf{H}^{\mathrm{p}}\right)$$

where $\mathbf{W}^{\mathrm{g}} \in \mathbb{R}^{l \times l}$ and $\mathbf{b}^{\mathrm{g}} \in \mathbb{R}^{l}$ are the learnable parameters. The outer product $(\cdot \otimes \mathbf{e}_Q)$ repeats the column vector $\mathbf{b}^{\mathrm{g}}$ Q times to form an $l \times l$ matrix. The i-th column of $\mathbf{G} \in \mathbb{R}^{Q \times P}$ represents the normalized attention weights over all the question words for the i-th word in passage.

We can use this attention matrix G to form representations of the question for each word in passage:

$$\overline{\mathbf{H}}^{\mathrm{q}} = \mathbf{H}^{\mathrm{q}}\mathbf{G} \tag{2}$$

Next, we produce the word matching representations $\mathbf{M} \in \mathbb{R}^{2l \times P}$ using $\mathbf{H}^{\mathrm{p}}$ and $\overline{\mathbf{H}}^{\mathrm{q}}$ as follows:

$$\mathbf{M} \quad = \quad \mathrm{ReLU}\left(\mathbf{W}^{\mathrm{m}}\begin{bmatrix}\mathbf{H}^{\mathrm{p}} \\ \overline{\mathbf{H}}^{\mathrm{q}} \\ \mathbf{H}^{\mathrm{p}} \bigodot \overline{\mathbf{H}}^{\mathrm{q}} \\ \mathbf{H}^{\mathrm{p}} - \overline{\mathbf{H}}^{\mathrm{q}}\end{bmatrix}\right), \tag{3}$$

where $\mathbf{W}^{\mathrm{m}} \in \mathbb{R}^{2l \times 4l}$ are learnable parameters; $\begin{bmatrix}\cdot \\ \cdot\end{bmatrix}$ is the column concatenation of matrices; Element-wise operations $(\cdot \bigodot \cdot)$ and $(\cdot - \cdot)$ are also used to represent word-level matching (Wang and Jiang 2017a; Chen et al. 2017b).

[4] Passage ranking models for non-factoid QA (Wang, Smith, and Mitamura 2007; Yang, Yih, and Meek 2015) are able to learn to rank these passages; but these models are trained using human annotated answer labels, which are not available here.

[5] In this paper we use sentence-level index thus each passage is an individual sentence. See the experimental setting.

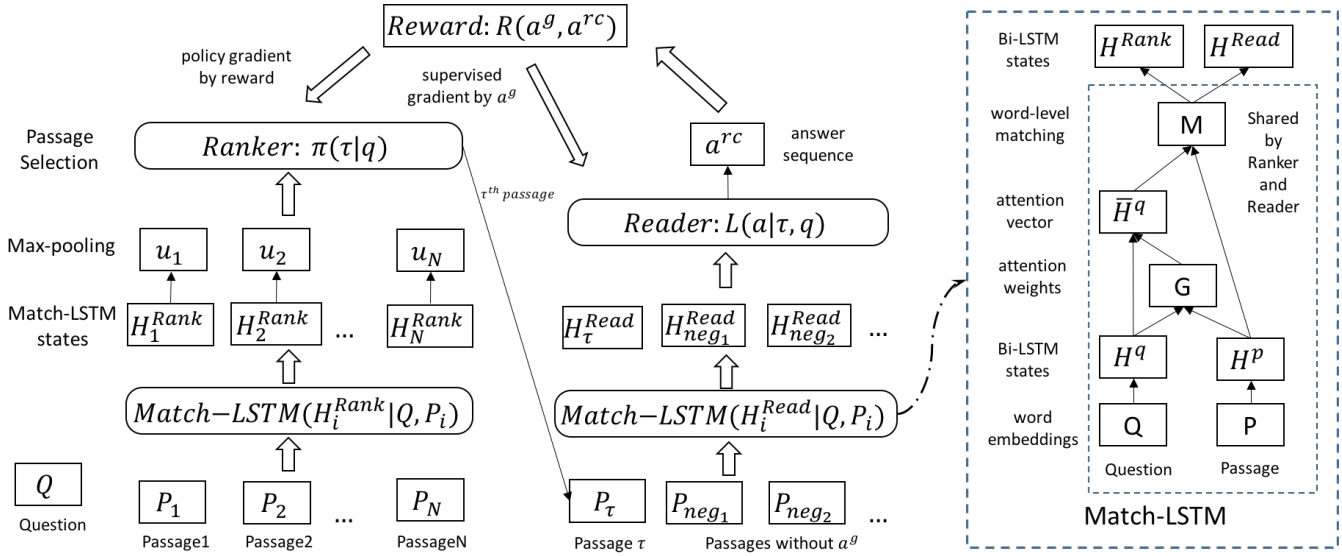

Figure 1: Overview of training our model, comprising a Ranker and a Reader based on Match-LSTM as shown on the right side. The Ranker selects a passage τ and the Reader predicts the start and end positions of the answer in τ. The reward for the Ranker depends on similarity of the extracted answer with the ground-truth answer $\mathbf{a}^g$. To accelerate Reader convergence, we also sample several negative passages without ground-truth answer.

Finally, we aggregate the word matching representations through another bi-directional LSTM:

$$\mathbf{H}^{\mathrm{m}} = \mathrm{BiLSTM}(\mathbf{M}), \tag{4}$$

where $\mathbf{H}^{\mathrm{m}} \in \mathbb{R}^{l \times P}$ is the sequence matching representation between a passage and a question.

To produce the input for the Ranker and Reader described next, we apply Match-LSTMs to the question and each of the passages. To reduce model complexity, the Ranker and Reader share the same $\mathbf{M}$ but have separate parameters for the aggregation stage shown in Eqn.(4), resulting different $\mathbf{H}^{\mathrm{m}}$, denoted as $\mathbf{H}^{\mathrm{Rank}}$ and $\mathbf{H}^{\mathrm{Read}}$ respectively.

Ranker Our Ranker selects passages for reading by the Reader. We train the Ranker using reinforcement learning, to output a policy or probability distribution over passages. First, we create a fixed-size vector representation for each passage from the matching representations $\mathbf{H}_{\mathrm{i}}^{\mathrm{Rank}}$, $\mathrm{i} \in [1, N]$, using a standard max pooling operation. The result $\mathbf{u}_{\mathrm{i}}$ is a representation of the i-th passage. We then concatenate the individual passage representations and apply a non-linear transformation followed by a normalization to compute the passage probabilities γ. Specifically:

$$\begin{aligned} \mathbf{u}_i &= \mathrm{MaxPooling}(\mathbf{H}_{\mathrm{i}}^{\mathrm{Rank}}), \\ \mathbf{C} &= \mathrm{Tanh}\left(\mathbf{W}^{\mathrm{c}}[\mathbf{u}_1; \mathbf{u}_2; ...; \mathbf{u}_N] + \mathbf{b}^{\mathrm{c}} \otimes \mathbf{e}_N\right), \\ \gamma &= \mathrm{Softmax}(\mathbf{w}^c\mathbf{C}), \end{aligned} \tag{5}$$

where $\mathbf{W}^{\mathrm{c}} \in \mathbb{R}^{l \times l}$ and $\mathbf{b}^{\mathrm{c}}, \mathbf{w}^{\mathrm{c}} \in \mathbb{R}^l$ are the parameters to optimize; $\mathbf{u}_i \in \mathbb{R}^l$ represents how the i^{th} passage matches the question; $\mathbf{C} \in \mathbb{R}^{l \times N}$ is a non-linear transformation of passage representations; and $\gamma \in \mathbb{R}^N$ is a vector of the predicted probabilities that each passage entails the answer.

The action policy is then defined as follows:

$$\pi(\tau|\mathbf{q}; \theta^r) = \gamma_\tau \tag{6}$$

where γ_τ is the probability of selecting passage τ, computed in Eqn.(5); θ^r represents parameters to learn. In the rest of the paper we denote the policy $\pi(\tau|q) = \pi(\tau|q; \theta^r)$ for simplicity. In this way, the action is to sample a passage according to its policy $\pi(\tau|q)$ as the input of Reader.

Reader Our Reader extracts an answer span from the passage τ selected by the Ranker. As in previous work (Wang and Jiang 2017b; Xiong, Zhong, and Socher 2017; Seo et al. 2017; Wang et al. 2017), the Reader is used to predict the start and end positions of the answer phrase in the passage.

First we process the output of Match-LSTMs on all the passages to produce the probability of the start position of the answer span β^{s}:

$$\begin{aligned} \mathbf{F}^{\mathrm{s}} &= \mathrm{Tanh}\left(\mathbf{W}^{\mathrm{s}}[\mathbf{H}_{\tau}^{\mathrm{Read}}; \mathbf{H}_{\mathrm{neg}_1}^{\mathrm{Read}}; ...; \mathbf{H}_{\mathrm{neg}_n}^{\mathrm{Read}}] + \mathbf{b}^{\mathrm{s}} \otimes \mathbf{e}_V\right), \\ \beta^{\mathrm{s}} &= \mathrm{Softmax}\left(\mathbf{w}^{\mathrm{s}}\mathbf{F}^{\mathrm{s}}\right), \end{aligned} \tag{7}$$

where neg_n is the id of a sampled passage not containing ground-truth answer during training; V is the total number of words in these passages; $\mathbf{e}_V$ is thus a V-dimension vector with ones; $[\cdot;\cdot]$ is the column concatenation operation; $\mathbf{W}^{\mathrm{s}} \in \mathbb{R}^{l \times l}$ and $\mathbf{b}^{\mathrm{s}}, \mathbf{w}^{\mathrm{s}} \in \mathbb{R}^l$ are the parameters to optimize; $\beta^{\mathrm{s}} \in \mathbb{R}^V$ is the probability of the start point of the span.

We similarly compute the probability of the ending position, $\beta^{\mathrm{e}} \in \mathbb{R}^V$, using separate parameters $\mathbf{W}^{\mathrm{e}}, \mathbf{b}^{\mathrm{e}}$ and $\mathbf{w}^{\mathrm{e}}$. The loss function can then be expressed as follows:

$$L(\mathbf{a}^g|\tau, \mathbf{q}) = -\log(\beta^{\mathrm{s}}_{a^s_\tau}) - \log(\beta^{\mathrm{e}}_{a^e_\tau}), \tag{8}$$

where $\mathbf{a}^g$ is the ground-truth answer; τ is sampled according to Eqn.(6), and during training, we keep sampling until passage τ contains $\mathbf{a}^g$; $\beta^s_{a^s_\tau}$ and $\beta^e_{a^e_\tau}$ represent the probability of the start and end positions of $\mathbf{a}^g$ in passage τ.

Training We adopt joint training of Ranker and Reader as shown in Algorithm 1. Since the Ranker makes a hard selection of the passage, it is trained using the REINFORCE algorithm. The Reader is trained using standard SGD/backprop.

Algorithm 1 Reinforced Ranker-Reader (R^3)

1: **Input:** $\mathbf{a}^g$, $\mathbf{q}$, passages from IR
2: **Output:** Θ
3: **Initialize:** $\Theta \leftarrow$ pre-trained Θ with a baseline method[6]
4: **for** each $\mathbf{q}$ in dataset **do**
5: For question $\mathbf{q}$, sample K passages from the top N passages retrieved by IR model for training. [7]
6: Randomly sample a positive passage $\tau \sim \pi(\tau|\mathbf{q})$
7: Extract the answer $\mathbf{a}^{rc}$ through RC model
8: Get reward r according to $R(\mathbf{a}^g, \mathbf{a}^{rc}|\tau)$.
9: Updating Ranker (ranking model) through policy gradient $r\frac{\partial}{\partial\Theta}\log(\pi(\tau|\mathbf{q}))$
10: Updating Reader (RC model) through supervised gradient $\frac{\partial}{\partial\Theta}L(\mathbf{a}^g|\tau,\mathbf{q})$
11: **end for**

Our training objective is to minimize the following loss function

$$J(\Theta) = -\mathbb{E}_{\tau\sim\pi(\tau|\mathbf{q})}\left[L(\mathbf{a}^g|\tau,\mathbf{q})\right], \tag{9}$$

where L is the loss of the Reader defined in Eqn. (8); $\pi(\tau|\mathbf{q})$ is the action policy defined in Eqn.(6); and Θ are parameters to be learned. During training, action sampling is limited solely to passages containing the ground-truth answer, to guarantee Reader updating (line 10 in Algorithm 1) based on the sampled passages with supervised gradients. The gradient of $J(\Theta)$ with respect to Θ is:

$$\begin{aligned}
\nabla_\Theta J(\Theta) &= -\nabla_\Theta \sum_\tau \pi(\tau|\mathbf{q})L(\mathbf{a}^g|\tau,\mathbf{q}) \\
&= -\sum_\tau \left(L(\mathbf{a}^g|\tau,\mathbf{q})\nabla_\Theta\pi(\tau|\mathbf{q}) + \pi(\tau|\mathbf{q})\nabla_\Theta L(\mathbf{a}^g|\tau,\mathbf{q})\right) \\
&= -\mathbb{E}_{\tau\sim\pi(\tau|\mathbf{q})}\left[L(\mathbf{a}^g|\tau,\mathbf{q})\nabla_\Theta\log(\pi(\tau|\mathbf{q}))\right. \\
&\qquad \left. + \nabla_\Theta L(\mathbf{a}^g|\tau,\mathbf{q})\right] \\
&\approx -\mathbb{E}_{\tau\sim\pi(\tau|\mathbf{q})}\left[R(\mathbf{a}^g,\mathbf{a}^{rc}|\tau)\nabla_\Theta\log(\pi(\tau|\mathbf{q}))\right. \\
&\qquad \left. + \nabla_\Theta L(\mathbf{a}^g|\tau,\mathbf{q})\right]
\end{aligned} \tag{10}$$

So in training, we first sample a passage τ according to the policy $\pi(\tau|\mathbf{q})$. Then the Reader updates its parameters given the passage τ using standard Backprop and the ranker updates its parameters via policy gradient using $L(a|\tau,\mathbf{q})$ as rewards. However, $L(a|\tau,\mathbf{q})$ is not bounded and introduces a large variance in gradients (similar to what was reported in Mnih et al. 2014). To address this, we replace $L(a|\tau,\mathbf{q})$ with a bounded reward $R(\mathbf{a}^g,\mathbf{a}^{rc}|\tau)$, which captures how well the answer extracted by the Reader matches the ground-truth answer. Specifically:

$$R(\mathbf{a}^g,\mathbf{a}^{rc}|\tau) = \begin{cases} 2, & \text{if } \mathbf{a}^g == \mathbf{a}^{rc} \\ f1(\mathbf{a}^g,\mathbf{a}^{rc}), & \text{else if } \mathbf{a}^g \cap \mathbf{a}^{rc} != \emptyset \\ -1, & \text{else} \end{cases} \tag{11}$$

where $\mathbf{a}^g$ is the ground-truth answer; $\mathbf{a}^{rc}$ is the answer extracted by Reader; $f1(\cdot,\cdot) \in [0,1]$ computes word-level F1 score between two sequences. F1 is used as reward when $\mathbf{a}^g$ and $\mathbf{a}^{rc}$ share some words but do not exactly match. We give a larger reward of 2 for exact match, and -1 reward for no overlap.

Prediction During testing, we combine the Ranker and Reader for answer extraction as follows:

$$\Pr(\mathbf{a},\tau) = \Pr(\mathbf{a}|\tau)\Pr(\tau) = e^{-L(\mathbf{a}|\tau,\mathbf{q})}\pi(\tau|\mathbf{q}), \tag{12}$$

where $\Pr(\mathbf{a},\tau)$ is the probability of extracting the answer $\mathbf{a}$ from passage τ. We select the answer with the largest $\Pr(\mathbf{a},\tau)$ as the final prediction.

Experimental Settings

To evaluate our model we have chosen five challenging datasets under the open-domain QA setting and three public baseline models.

Datasets

We experiment with five different datasets whose statistics are shown in Table 2.

Quasar-T is a dataset for SR-QA, with question-answer pairs from various internet sources. Each question is compared to 100 sentence-level candidate passages, retrieved by their IR model from the ClueWeb09 data source, to extract the answer.

The other four datasets we consider are: **SQuAD**, the Stanford QA dataset, from which we take only the question-answer pairs and discard the passages to form an open-domain QA setting (denoted as **SQuAD$_{\text{OPEN}}$**); **WikiMovies** which contains movie-related questions from the OMDb and MovieLens databases and where the questions can be answered using Wikipedia pages; **CuratedTREC**, based on TREC (Voorhees and Tice 2000) and designed for open-domain QA; and **WebQuestion** which is designed for knowledge-base QA with answers restricted to Freebase entities. For these four datasets under the open-domain QA setting, no candidate passages are provided so we build a similar sentence-level Search Index based on English Wikipedia, following Chen et al. 2017a's work. To provide a small yet sufficient search space for our model, we employ a traditional IR method to retrieve relevant passages from the whole of Wikipedia. We use the 2016-12-21 dump of English Wikipedia as our sole knowledge source, and build an

[6]Baseline method SR^2, described in Experimental Settings.

[7]For computational efficency, we sample 10 passages during training, and make sure there are at least 2 negative passages and as many positive passages as possible.

	#q(train)	#q(test)	#p(train)	#p(test)
Quasar-T	28496	3000	14.8 / 100	1.9 / 50
$\text{SQuAD}_{\text{OPEN}}$	82271	10570	35.1 / 200	2.3 / 50
WikiMovies	93935	9,952	68.5 / 200	1.8 / 50
CuratedTREC	1204	694	14.6 / 200	4.8 / 50
WebQuestion	3272	2,032	57.2 / 200	4.1 / 50

Table 2: Statistics of the datasets. #q represents the number of questions. For the training dataset, we ignore the questions without any answer in all the retrieved passages. In the special case that there's only one answer for the question, during training, we combine the question with the answer as the query to improve IR recall. Otherwise we use only the question. #p represents the number of passages and 14.8 / 100 means there are 14.8 passages containing the answer on average out of the 100 passages. We use top50 passages retrieved by the IR model for testing.

inverted index with Lucene[8]. We then take each input question as a query to search for top-200 articles, rank them with BM25, and split them into sentences. The sentences are then ranked by TF-IDF and the top-200 sentences for each question retained.

Baselines

We consider three public baseline models[9]: GA (Dhingra et al. 2017; Dhingra, Mazaitis, and Cohen 2017), a gated-attention reader for text comprehension; BiDAF (Seo et al. 2017), a reader with bidirectional attention flow for machine comprehension; and DrQA (Chen et al. 2017a), a document reader for question answering. We also compare our model R^3 with two internal baselines:

Single Reader (SR) This model is trained in the same way as Chen et al. 2017a and Dhingra, Mazaitis, and Cohen 2017. We find all the answer spans that exactly match the ground-truth answers from the retrieved passages and train the Reader using the objective of Eqn.(8). Here τ is randomly sampled from $[1, N]$ instead of using Eqn.(6).

Simple Ranker-Reader (SR^2) This Ranker-Reader model is trained by combining the two different objective functions for the Single Reader and the Ranker models together. In order to train the Ranker, we treat all the passages that contain the ground-truth answer as positive cases and use the following for the Ranker loss:

$$\sum_{n=1}^{N} y_n \left(\log(y_n) - \log(\gamma_n)\right), \tag{13}$$

which is the KL divergence between γ computed through Eqn.(5) and a probability vector $\mathbf{y}$, where $y_i = 1/N_p$ when the passage i contains the ground-truth answer, and $y_i = 0/N_p$ otherwise. N_p is the total number of passages which contain the ground-truth answer in the top-N passage list.

Implementation Details

In order to increase the likelihood that question-related context will be contained in the retrieved passages for the training dataset, if the answer is unique, we combine the question with the answer to form the query for information retrieval. For the testing dataset, we use only the question as a query and collect the top 50 passages for answer extraction.

During training, our R^3 model is first initialized by pre-training the model using the Simple Ranker-Reader (R^2), to encourage convergence. As discussed earlier, the pre-processing and matching layers, Eqn.(1-3), are shared by both Ranker and Reader. The number of LSTM layers in Eqn.(4) is set to 3 for the Reader and 1 for the Ranker.

Our model is optimized using Adamax (Kingma and Ba 2015). We use fixed GloVe (Pennington, Socher, and Manning 2014) word embeddings. We set l to 300, batch size to 30, learning rate to 0.002 and tune the dropout probability.

Results and Analysis

In this section, we will show the performance of different models on five QA datasets and offer further analysis.

Overall Results

Our results are shown in Table 3. We use F1 score and Exact Match (EM) evaluation metrics[10]. We first observe that on Quasar-T, the Single Reader can exceed state-of-the-art performance. Moreover, unlike DrQA, our models are all trained using distant supervision and, without pre-training on the original SQuAD dataset[11], our Single Reader model still achieves better performance on the WikiMovie and CuratedTREC datasets.

Next we observe that the Reinforced Ranker-Reader (R^3) achieves the best performance on the Quasar-T, WikiMovies, and CuratedTREC datests and achieves significantly better performance than our internal baseline model Simple Ranker-Reader (SR^2) on all datasets except CuratedTREC. These results demonstrate the effectiveness of using RL to jointly train the Ranker and Reader both as compared to competing approaches and the non-RL Ranker-Reader baseline.

Further Analysis

In this subsection, we first present an analysis of the improvement of both Ranker and Reader trained with our method, and then discuss ideas for further improvement.

Quantitative Analysis First, we examine whether our RL approach could help the Ranker overcome the absence of any ground-truth ranking score. To control everything but the change in Ranker, we conduct two experiments combining the same Single Reader with two different Rankers

[8] `https://lucene.apache.org/`

[9] We only compare to the results from the public papers.

[10] Evaluation tooling is from SQuAD (Rajpurkar et al. 2016).

[11] The performance of our Single Reader model on the original SQuAD dev set is F1 77.0, EM 67.6 which is close to the BiDAF model, F1 77.3, EM 67.7 and DrQA model, F1 78.8, EM 69.5.

	Quasar-T		$\text{SQuAD}_{\text{OPEN}}$		WikiMovies		CuratedTREC		WebQuestions	
	F1	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**	**EM**
GA (Dhingra et al. 2017)	26.4	26.4	-	-	-	-	-	-	-	-
BiDAF (Seo et al. 2017)	28.5	25.9	-	-	-	-	-	-	-	-
DrQA (Chen et al. 2017a)	-	-	-	28.4	-	34.3	-	25.7	-	**19.5**
Single Reader (SR)	$38.5^{.2}$	$31.5^{.2}$	$35.4^{.2}$	$26.9^{.2}$	$38.8^{.1}$	$37.7^{.1}$	$33.6^{.6}$	$27.4^{.4}$	$22.0^{.2}$	$15.2^{.3}$
Simple Ranker-Reader (SR^2)	$38.8^{.2}$	$31.9^{.2}$	$35.8^{.2}$	$27.2^{.2}$	$39.3^{.1}$	$38.1^{.1}$	$33.4^{.6}$	$27.7^{.5}$	$22.5^{.3}$	$15.6^{.4}$
Reinforced Ranker-Reader (R^3)	$\mathbf{40.9}^{.3}$	$\mathbf{34.2}^{.3}$	$\mathbf{37.5}^{.2}$	$\mathbf{29.1}^{.2}$	$\mathbf{39.9}^{.1}$	$\mathbf{38.8}^{.1}$	$\mathbf{34.3}^{.6}$	$\mathbf{28.4}^{.6}$	$\mathbf{24.6}^{.3}$	$17.1^{.3}$
DrQA-MTL (Chen et al. 2017a)	-	-	-	29.8	-	36.5	-	25.4	-	20.7
YodaQA (Baudiš and Šedivỳ 2015)	-	-	-	-	-	-	-	31.3	-	39.8

Table 3: Open-domain question answering results. The results show the average of 5 runs, with standard error in the superscript. The CuratedTREC and WebQuestions models are initialized by training on $\text{SQuAD}_{\text{OPEN}}$ first. On the bottom, YodaQA and DrQA-MTL use additional resources (usage of KB for the former, and multiple training datasets for the latter), so are not a true apple-to-apple comparison to the other methods. EM: Exact Match.

	F1	EM
Single Reader (SR)	38.3	31.4
SR + Ranker (from SR^2)	38.9	31.8
SR + Ranker (from R^3)	**40.0**	**33.1**
SR^2	38.7	31.9
R^3	40.8	34.1

Table 4: Effects of rankers from SR^2 and R^3 (on Quasar-T test dataset). Here we use the same single reader model (SR) as the reader, combined with two different rankers. The performance of the two runs of SR^2 and R^3 (that provide the rankers) is listed at bottom.

	TOP-k	F1	EM
Single Reader (SR)	1	38.3	31.4
Single Reader (SR)	3	51.7	43.7
Single Reader (SR)	5	58.7	49.2
SR + Ranker (from R^3)	1	**40.0**	**33.1**

Table 5: Potential improvement on QA performance by improving the ranker. The performance is based on the Quasar-T test dataset. The **TOP-3/5** performance is used to evaluate the further potential improvement by improving rankers (see the "Potential Improvement" section).

trained from SR^2 and R^3, respectively. Table 4 shows the results on the Quasar-T test dataset. Note that the Single Reader combined with the Ranker trained from R^3 model achieves an EM 1.3 higher performance than combined with the Ranker from SR^2 which treats all passages containing ground-truth answer as positive cases. That means our proposed Ranker is better than the Ranker normally trained in the distant supervision setting.

We also find that the performance of R^3 can still achieve an EM 1.0 higher than the Single Reader combined with the Ranker from R^3 through Table 4. In this setting, the Ranker is the same, while the Reader is trained differently. We infer from this that our proposed methods R^3 can not only improve the Ranker but also the Reader.

Potential Improvement We offer a statistical analysis to approximate the upper bound achievable by only improving the ranking models. This is evaluated by computing the QA performance with the best passage among the top-k ranked passages. Specifically, for each question, we extract one answer from each of the top-50 passages retrieved from the IR system, and take the top-k answers with the highest scores according to Eqn.(12) from these. Based on the k answer candidates, we compute the **TOP-k** F1/EM by evaluating on the answer with highest F1/EM score for each question. This is equivalent to having an *oracle ranker* that assigns a $+\infty$ score to the passage (from the passages providing top-k candidates) yielding the best answer candidate.

Table 5 shows a clear gap between TOP-3/5 and TOP-1 QA performances (over 12-20%). According to our evaluation approach of TOP-k F1/EM and since the same SR model is used, this gap is solely due to the oracle ranker. Although our model is far from the oracle performance, it still provides a useful upper bound for improvement.

Ranker Performance Analysis Next we show the intermediate performance of our method on the ranking step. Since we do not have the ground-truth for the ranking task, we evaluate on pseudo labels: a passage is considered positive if it contains the ground-truth answer. Then a ranker's top-k output is considered accurate if any of the k passages contain the answer (i.e. *top-k recall*). Note that this way of evaluation on top-1 is consistent with the training objective of the ranker in SR^2.

From the results in Table 7, the Ranker from R^3 performs significantly better than the one from SR^2 on top-1 and top-3 performance, despite the fact that it is not directly trained to optimize this pseudo accuracy. Given the evaluation bias that favors the SR^2, this indicates that our R^3 model could make Ranker training easier, compared to training on the objective in Eqn.13 with pseudo labels.

Starting from top-5, the Ranker from R^3 gives slightly lower recall. This is because the two Rankers have a simi-

Q	Apart from man what is New Zealand 's only native mammals	
A	**bats**	
	Reinforced Ranker-Reader (R^3)	Simple Ranker-Reader (SR^2)
P1	New Zealand has no native land mammals apart from some rare **bats** .	New Zealand 's native species were **sitting ducks** !
P2	New Zealand 's native species were sitting **ducks** !	1080 is a commonly used pesticide since it is very effective on mammals and New Zealand has no native land mammals apart from two species of **bat** .
P3	-LSB- edit -RSB- Fauna **Bats** were the only mammals of New Zealand until the arrival of humans .	Previously it had been thought that **bats** were the only terrestrial mammals native to New Zealand .

Table 6: An example of the answers extracted by the R^3 and SR^2 methods, given the question. The words in bold are the extracted answers. The passages are ranked by the highest score (Ranker+Reader) of the answer span in each passage.

	TOP-1	TOP-3	TOP-5
IR	19.7	36.3	44.3
Ranker from SR^2	28.8	46.4	54.9
Ranker from R^3	40.3	51.3	54.5

Table 7: The performance of Rankers (recall of the top-k ranked passages) on the Quasar-T test dataset. This evaluation is simply based on whether the ground-truth appears in the TOP-N passages. IR directly uses the ranking score from raw dataset.

lar ability to rank the potentially useful passages in the top-5, but the evaluation bias benefits the SR^2 Ranker. Overall, our R^3 could successfully rank the potentially more useful passages to the highest positions (top 1-3), improving the overall QA performance.

An example in Table 6 illustrates the importance of ranking. The passages on the left are from the R^3 Ranker and the ones on the right from the SR^2 Ranker. If SR^2 ranked P2 or P3 higher, it could also have extracted the right answer. In general, if passages that can entail the answer are ranked more accurately, both models could be improved.

Related Work

Open domain question answering dates back to as early as (Green Jr et al. 1961) and was popularized with TREC-8 (Voorhees 1999). The task is to answer a question by exploiting resources such as documents (Voorhees 1999), webpages (Kwok, Etzioni, and Weld 2001; Chen and Van Durme 2017) or structured knowledge bases (Berant et al. 2013; Bordes et al. 2015; Yu et al. 2017). An early consensus since TREC-8 has produced an approach with three major components: question analysis, document retrieval and ranking, and answer extraction. Although question analysis is relatively mature, answer extraction and document ranking still represent significant challenges.

Very recently, IR plus machine reading comprehension (**SR-QA**) showed promise for open-domain QA, especially after datasets created specifically for the multiple-passage RC setting (Nguyen et al. 2016; Chen et al. 2017a; Joshi et al. 2017; Dunn et al. 2017; Dhingra, Mazaitis, and Cohen 2017). These datasets deal with the end-to-end open-domain QA setting, where only question-answer pairs provide supervision. Similarly to previous work on open-domain QA, existing deep learning based solutions to the above datasets also rely on a document retrieval module to retrieve a list of passages for RC models to extract answers. Therefore, these approaches suffer from the limitation that the passage ranking scores are determined by n-gram matching (with tf-idf weighting), which is not ideal for QA.

Our ranker module in R^3 could help to alleviate the above problem, and RL is a natural fit to jointly train the ranker and reader since the passages do not have ground-truth labels. Our work is related to the idea of soft or hard attentions (usually with reinforcement learning) for hierarchical or coarse-to-fine decision sequences making in NLP, where the attentions themselves are latent variables. For example, Lei, Barzilay, and Jaakkola 2016 propose to first extract informative text fragments then feed them to text classification and question retrieval models. Cheng and Lapata 2016 and Choi et al. 2017 proposed coarse-to-fine frameworks with an additional sentence selection step before the original word-level prediction for text summarization and reading comprehension, respectively. To the best of our knowledge, we are the first apply this kind of framework to the open-domain question answering.

From the method-perspective, our work is most close to Choi et al. 2017's work in terms of the usage of REINFORCE. Our main aim is to deal with the lack of annotation in the passage selection step, which is a necessary intermediate step in open-domain QA. In comparison, Choi et al. 2017 has as its main aim to speed up the RC model in the single passage setting. From the motivation-perspective, we are similar to Narasimhan, Yala, and Barzilay 2016's work. Both work aim to find passages easy and suitable for the QA or IE models to extract answers, in order to boost accuracy.

Conclusion

We have proposed and evaluated R^3, a new open-domain QA framework which combines IR with a deep learning based Ranker and Reader. First the IR model retrieves the top-N passages conditioned on the question. Then the Ranker and Reader are trained jointly using reinforcement learning to directly optimize the expectation of extracting the ground-truth answer from the retrieved passages. Our framework achieves the best performance on several QA datasets.

References

[2015] Baudiš, P., and Šedivỳ, J. 2015. Modeling of the question answering task in the yodaqa system. In *Intl. Conf. of the Cross-Language Evaluation Forum for European Languages*.

[2013] Berant, J.; Chou, A.; Frostig, R.; and Liang, P. 2013. Semantic parsing on freebase from question-answer pairs. In *Proc. of Conf. on EMNLP*.

[2015] Bordes, A.; Usunier, N.; Chopra, S.; and Weston, J. 2015. Large-scale simple question answering with memory networks. In *Proc. of ICLR*.

[2017] Chen, T., and Van Durme, B. 2017. Discriminative information retrieval for question answering sentence selection. In *Proc. of Conf. on EACL*.

[2017a] Chen, D.; Fisch, A.; Weston, J.; and Bordes, A. 2017a. Reading Wikipedia to answer open-domain questions. In *Proc. of ACL*.

[2017b] Chen, Q.; Zhu, X.; Ling, Z.; Wei, S.; Jiang, H.; and Inkpen, D. 2017b. Enhanced lstm for natural language inference. In *Proc. of ACL*.

[2016] Cheng, J., and Lapata, M. 2016. Neural summarization by extracting sentences and words. In *Proc. of ACL*.

[2017] Choi, E.; Hewlett, D.; Uszkoreit, J.; Polosukhin, I.; Lacoste, A.; and Berant, J. 2017. Coarse-to-fine question answering for long documents. In *Proc. of ACL*.

[2017] Dhingra, B.; Liu, H.; Yang, Z.; Cohen, W. W.; and Salakhutdinov, R. 2017. Gated-attention readers for text comprehension. In *Proc. of ACL*.

[2017] Dhingra, B.; Mazaitis, K.; and Cohen, W. W. 2017. QUASAR: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904*.

[2017] Dunn, M.; Sagun, L.; Higgins, M.; Guney, U.; Cirik, V.; and Cho, K. 2017. SearchQA: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*.

[2010] Ferrucci, D.; Brown, E.; Chu-Carroll, J.; Fan, J.; Gondek, D.; Kalyanpur, A. A.; et al. 2010. Building watson: An overview of the deepqa project. *AI magazine* 31(3):59–79.

[1961] Green Jr, B. F.; Wolf, A. K.; Chomsky, C.; and Laughery, K. 1961. Baseball: an automatic question-answerer. In *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer Conf.*, 219–224. ACM.

[2017] Joshi, M.; Choi, E.; Weld, D. S.; and Zettlemoyer, L. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proc. of ACL*.

[2015] Kingma, D., and Ba, J. 2015. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

[2001] Kwok, C.; Etzioni, O.; and Weld, D. S. 2001. Scaling question answering to the web. *ACM Transactions on Information Systems (TOIS)* 19(3):242–262.

[2016] Lei, T.; Barzilay, R.; and Jaakkola, T. 2016. Rationalizing neural predictions. In *Proc. of Conf. on EMNLP*.

[2014] Mnih, V.; Heess, N.; Graves, A.; et al. 2014. Recurrent models of visual attention. In *Advances in NIPS*, 2204–2212.

[2016] Narasimhan, K.; Yala, A.; and Barzilay, R. 2016. Improving information extraction by acquiring external evidence with reinforcement learning. In *Proc. of EMNLP*.

[2016] Nguyen, T.; Rosenberg, M.; Song, X.; Gao, J.; Tiwary, S.; Majumder, R.; and Deng, L. 2016. MS MARCO: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

[2014] Pennington, J.; Socher, R.; and Manning, C. D. 2014. GloVe: Global vectors for word representation. In *Proc. of Conf. on EMNLP*.

[2016] Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proc. of Conf. on EMNLP*.

[2017] Seo, M.; Kembhavi, A.; Farhadi, A.; and Hajishirzi, H. 2017. Bidirectional attention flow for machine comprehension. In *Proc. of ICLR*.

[2000] Voorhees, E. M., and Tice, D. M. 2000. Building a question answering test collection. In *Proc. of 23rd annual Intl. ACM SIGIR Conf. on Research and development in information retrieval*, 200–207. ACM.

[1999] Voorhees, E. M. 1999. The trec-8 question answering track report. In *Trec*, volume 99, 77–82.

[2016] Wang, S., and Jiang, J. 2016. Learning natural language inference with LSTM. In *Proc. of Conf. on NAACL*.

[2017a] Wang, S., and Jiang, J. 2017a. A compare-aggregate model for matching text sequences. In *Proc. of ICLR*.

[2017b] Wang, S., and Jiang, J. 2017b. Machine comprehension using match-LSTM and answer pointer. In *Proc. of ICLR*.

[2016] Wang, Z.; Mi, H.; Hamza, W.; and Florian, R. 2016. Multi-perspective context matching for machine comprehension. *arXiv preprint arXiv:1612.04211*.

[2017] Wang, W.; Yang, N.; Wei, F.; Chang, B.; and Zhou, M. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proc. of ACL*.

[2007] Wang, M.; Smith, N. A.; and Mitamura, T. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *Proc. of Conf. on EMNLP*.

[1992] Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*.

[2017] Xiong, C.; Zhong, V.; and Socher, R. 2017. Dynamic coattention networks for question answering. In *Proc. of ICLR*.

[2015] Yang, Y.; Yih, W.-t.; and Meek, C. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proc. of Conf. on EMNLP*.

[2017] Yu, M.; Yin, W.; Hasan, K. S.; Santos, C. d.; Xiang, B.; and Zhou, B. 2017. Improved neural relation detection for knowledge base question answering. In *Proc. of ACL*.