A Framework for Automatic Question Generation from Text using Deep Reinforcement Learning

Vishwajeet Kumar
IITB-Monash Research Academy
Mumbai, India
vishwajeet@cse.iitb.ac.in

Ganesh Ramakrishnan
IIT Bombay
Mumbai India
ganesh@cse.iitb.ac.in

Yuan-Fang Li
Faculty of Information Technology
Monash University, Australia
yuanfang.li@monash.edu

arXiv:1808.04961v1 [cs.CL] 15 Aug 2018

Abstract

Automatic question generation (QG) is a useful yet challenging task in NLP. Recent neural network-based approaches represent the state-of-the-art in this task, but they are not without shortcomings. Firstly, these models lack the ability to handle rare words and the word repetition problem. Moreover, all previous works optimize the cross-entropy loss, which can induce inconsistencies between training (objective) and testing (evaluation measure). In this paper, we present a novel deep reinforcement learning based framework for automatic question generation. The generator of the framework is a sequence-to-sequence model, enhanced with the copy mechanism to handle the rare-words problem and the coverage mechanism to solve the word repetition problem. The evaluator model of the framework evaluates and assigns a reward to each predicted question. The overall model is trained by learning the parameters of the generator network which maximizes the reward. Our framework allows us to directly optimize any task-specific score including evaluation measures such as BLEU, GLEU, ROUGE-L, *etc.*, suitable for sequence to sequence tasks such as QG.
Our comprehensive evaluation shows that our approach significantly outperforms state-of-the-art systems on the widely-used SQuAD benchmark in both automatic and human evaluation.

1 Introduction

Automatic question generation (QG) is a very important yet challenging problem in NLP. It is defined as the task of generating syntactically sound, semantically correct and relevant questions from various input formats such as text, a structured database or a knowledge base (Mannem et al., 2010). Question generation can be naturally applied in many domains such as MOOC, automated help systems, search engines, chatbot systems (e.g. for customer interaction), and healthcare for analyzing mental health (Crangle and Kart, 2015).

Despite its usefulness, manually creating meaningful and relevant questions is a time-consuming and challenging task. For example, while evaluating students on reading comprehension, it is tedious for a teacher to manually create questions, find answers to those questions, and thereafter evaluate answers.

Traditional approaches have either used a linguistically motivated set of transformation rules for transforming a given sentence into a question or a set of manually created templates with slot fillers to generate questions. Recently, neural network based techniques such as sequence-to-sequence (Seq2Seq) learning have achieved remarkable success in various NLP tasks, including question generation. A recent deep learning approach to question generation by Serban et al. (2016) investigates a simpler task of generating questions only from a triplet of subject, relation and object. Du et al. (2017) propose a Seq2Seq model with attention for question generation from text. Kumar et al. (2018) generate question-answer pairs from text using Pointer Networks (Vinyals et al., 2015).

We motivate our work by identifying limitations of prior work, illustrated on the example sentence at the top of Table 1. We identify three main limitations, namely the inadequate loss function, the rare word problem, and word repetition problem.

A Seq2Seq model trained using a vanilla (decomposable) cross-entropy loss function generates the question *"what year was new york named ?"* (row 1 in Table 1), which is not addressed in the sentence. The passage talks only about founding of the city and its naming 2 years later. This is possibly because of the use of a loss that is somewhat agnostic to sequence information. Given its decomposable nature, the cross-entropy loss on the ground truth question or any of its (syntactically invalid) anagrams will be the same.

Using cross-entropy loss in sequence prediction could make the process brittle, since the model trained on a specific distribution over words is used on a test dataset with a possibly different distribution to predict

Text: "new york city traces its roots to its 1624 founding as a trading post by colonists of the dutch republic and was named new amsterdam in 1626 ."		
Row	**Model**	**Question generated**
1	Seq2Seq model optimized on vanilla (cross entropy) loss	what year was new york named ?
2	Seq2Seq model optimized on BLEU (using RL)	what year was new york founded ?
3	Copy aware Seq2Seq model optimized on BLEU (using RL)	what year was new new amsterdam named ?
4	Coverage and Copy aware Seq2Seq model optimized on BLEU (using RL)	in what year was new amsterdam named ?

Table 1: Sample text and questions generated using variants of our model.

the next word given the current predicted word. This creates exposure bias (Ranzato et al., 2015), since during training, the model is only exposed to the data distribution and not the model distribution.

The standard metrics for evaluating the performance of question generation models such as BLEU (Papineni et al., 2002), GLEU, and ROUGE-L (Lin, 2004) are based on degree of n-gram overlaps between a generated question and the ground-truth question. Reinforcement learning (Sutton and Barto, 1998) (RL) allows us to use policy gradient to directly optimize task-specific rewards (such as BLEU, GLEU and ROUGE-L), which are otherwise non-differentiable and hard to optimize. Row 2 in Table 1 is a significantly better question generated by accounting for sequence information in the loss, that accounts for proximity of 'founded' to 'new york'.

We recall that the passage also dealt with the naming of the city as 'new amsterdam'. Since 'new amsterdam' itself is much less frequent than 'new york', it is highly unlikely for a decoder that is solely based on a language model to generate such a word with very rare occurrences in a corpus. In such cases, the possibly rare words in the input sentence might be required to be *copied* from the sentence to the question. We incorporate a *copy* mechanism to handle the rare word problem. Row 3 in Table 1 illustrates the influence of our copy mechanism (described later), where a rare phrase 'new amsterdam' has been rightly picked up in association with the name of the city. However, note that the word 'new' has been meaninglessly repeated twice.

Since the encoder-decoder based model could generate questions with meaningless repetitions, we introduce a mechanism for discouraging repetitions by quantitatively emphasizing the *coverage* of sentence words while decoding. In row 4 of Table 1, we present the result of our RL model trained by incorporating both the copy and coverage mechanisms. As can be seen, the generated question is syntactically and semantically correct, and relevant to the passage. We present one more example later in Table 5 and several examples in Section 2 of the supplementary material.

In this paper, we present a novel approach that addresses the challenges described above. Our main contributions are threefold: (i) a novel *deep reinforcement learning based framework* that allows the direct optimization of task-specific metrics; (ii) a novel loss function that incorporates both the cross-entropy loss and the reinforcement learning loss; and (iii) a novel technique making use of the *copy* mechanism to handle the rare word problem, and the *coverage* mechanism to handle the word repetition problem.

When evaluated on the benchmark SQuAD dataset (Rajpurkar et al., 2016), our system considerably outperforms current state-of-the art models in automatic question generation (Du et al., 2017; Kumar et al., 2018) in both automatic and human evaluation.

2 Problem Formulation

Automatic question generation from text can be formulated as a sequence-to-sequence learning problem. Given a text $\mathbf{X}$ as a sequence of words, the goal is to generate a question $\mathbf{Q}$, which is syntactically and semantically correct, meaningful, relevant and natural. More specifically, the main objective is to learn the underlying conditional probability distribution $P_\theta(\mathbf{Q}|\mathbf{X})$ parameterized by θ. In other words, the aim is to learn a model θ during training using text-question pairs such that the probability $P_\theta(\mathbf{Q}|\mathbf{X})$ is maximized over the given training dataset.[1]

More formally let us represent the text $\mathbf{X}$ as a sequence of words $(x_1,x_2,x_3,...,x_M)$ of length M, and a question $\mathbf{Q}$ as a sequence of words $(y_1,y_2,y_3,...,y_L)$ of length L. Mathematically, the model is meant to generate $\mathbf{Q^*}$ such that $\mathbf{Q^*} = \arg\max_{\mathbf{Q}} P_\theta(\mathbf{Q}|\mathbf{X})$ where $P_\theta(\mathbf{Q}|\mathbf{X})$ is defined as:

$$P_\theta(\mathbf{Q}|\mathbf{X}) = \prod_{i=1}^{L} P_\theta(y_i|y_1,...,y_{i-1};x_1,...,x_M) \quad (1)$$

Typically, such a model is trained by optimizing the cross-entropy loss. As motivated earlier, such a loss is sensitive to decoding-time errors. Therefore, we

[1] Notations used in the rest of the paper are also summarized in Section 1 of the supplementary material.

posit that it is more appropriate to directly optimize an evaluation function such as BLEU or ROUGE-L, which are more suitable to the question generation task. We refine this maximum likelihood based objective using deep reinforcement learning.

Our reinforcement learning (RL) framework for question generation consists of a generator and an evaluator. We refer to the generator as the *agent* and to the generation of the next word as an *action*. The probability of decoding a word $P_\theta(word)$ gives a stochastic *policy*. On every token that is output, an evaluator assigns a reward for the output sequence predicted so far using the current policy of the generator. Based on the reward assigned by the evaluator, the generator updates and improves its current policy. Let us denote the reward (*return*) at time step t by r_t. The cumulative reward, computed at the end of the sequence generated is represented by $R = \sum_{t=0}^{T} r_t$. The goal in RL-based question generation is to determine a generator (policy) that maximizes the expected return:

$$Loss_{RL}(\theta) = \mathrm{E}_{P_\theta(y_{0:T}|\mathbf{X})} \sum_{t=0}^{T} r_t(y_t; X, Y_{0:t-1}) \quad (2)$$

where X is the current input, $Y_{0:t-1}$ is the predicted sequence until time $t-1$. This supervised learning framework allows us to directly optimize task-specific evaluation metrics (r_t) such as BLEU.

3 Our Approach

The two main components of our framework are a generator and an evaluator. The generator is a sequence-to-sequence model, augmented with (i) the copy mechanism (Gu et al., 2016) to handle rare words, and (ii) the coverage mechanism (Tu et al., 2016) to discourage word repetitions. The evaluator provides rewards to fine-tune the generator. The reward function can be chosen to be a combination of one or more metrics. The high-level architecture of our question generation framework is presented in Figure 1.

3.1 Generator

The generator is a sequence-to-sequence model that uses attention and incorporates the copy and coverage mechanisms. It takes as input a sequence of words augmented with a set of linguistic features and at each step, outputs a word with the highest probability, to eventually produce a word sequence.

3.1.1 Baseline sequence-to-sequence model with attention

Sentence Encoder: Each word in the input text is fed sequentially along with linguistic features into the encoder, which generates a sequence of hidden states. Our encoder is a two-layer bidirectional LSTM network,

$$\overrightarrow{fw_t} = \overrightarrow{LSTM_2}(x_t, \overrightarrow{h_{t-1}})$$
$$\overleftarrow{bw_t} = \overleftarrow{LSTM_2}(x_t, \overleftarrow{h_{t-1}})$$

where x_t is the given input word at time step t, and $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ are the hidden states at time step t for the forward pass and backward pass respectively. The hidden states (from the forward and backward pass) of the last layer of the sentence encoder are concatenated to form state s as $s = [\overrightarrow{fw_t}; \overleftarrow{bw_t}]$.

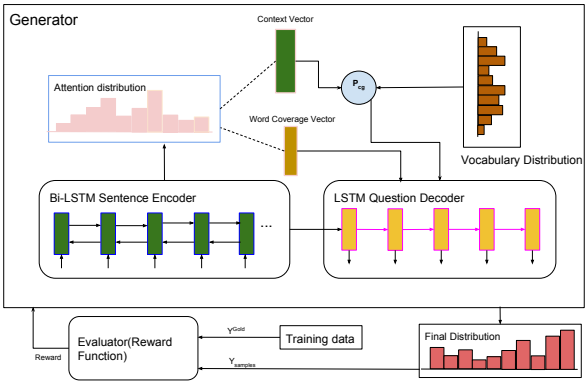

Figure 1: Our deep Reinforcement Learning framework for question generation. p_{cg} is the probability which determines whether to copy a word from source text or sample it from vocabulary distribution.

Question Decoder: Our question decoder is a single-layer LSTM network, initialized with the state $s = [\overrightarrow{fw_t}; \overleftarrow{bw_t}]$. Let $qword_t$ be the target word at time stamp t as per the ground truth question. During training, at each time step t the decoder takes as input the embedding vector of the previous word $qword_{t-1}$, concatenated with embedding of features of $qword_{t-1}$ to produce the decoder hidden state (s_t). The decoder produces one symbol at a time and stops when the [STOP] symbol is emitted. The only change with the decoder at test time is that it uses output from the previous word emitted by the decoder in place of $word_{t-1}$ (since there is no access to ground truth then).

We also model the attention (Bahdanau et al., 2014) distribution over words in the source text. We calculate the attention (a_i^t) over the i^{th} source word as

$$e_i^t = v^t tanh(W_{eh} h_i + W_{sh} s_t + b_{att}) \quad (3)$$
$$a_i^t = softmax(e_i^t) \quad (4)$$

Here v^t, W_{eh}, W_{sh} and b_{att} are model parameters to be learned. And h_i is the concatenation of forward and backward hidden states of the encoder. We use

this attention a_i^t to generate the *context vector* c_t^* as a weighted sum of encoder hidden states: $c_t^* = \sum_i a_i^t h_i$. We further use the c_t^* vector to obtain a probability distribution over the words in the vocabulary as: $P = softmax(W_v[s_t, c_t^*] + b_v)$, where W_v and b_v are model parameters. Thus during decoding, the probability of a word is $P(qword)$. During the training process for each timestamp, the loss is calculated as $L_t = -\log P(qword_t)$. The loss associated with the generated question is:

$$Loss = \frac{1}{T}\sum_{t=0}^{T} L_t = -\frac{1}{T}\sum_{t=0}^{T} \log P(qword_t) \quad (5)$$

3.1.2 Adding Copy and Coverage Mechanism

Recap that the copy mechanism was to facilitate copying some entities and words from the source sentence to the question. We describe this mechanism more specifically through another example sentence: "*The captain of Indian cricket team is Virat Kohli*". In the generation of a meaningful question "*Who is the captain of Indian cricket team?*", the words *the*, *is*, *captain*, *of*, *Indian*, *cricket*, and *team* are directly copied, whereas the word *who* is generated by the decoder based on the probability distribution over the training vocabulary. So in order to learn to copy (from source) as well as to generate words from the vocabulary (using the decoder), we calculate $p_{cg} \in [0,1]$ as the decision of a binary classifier that determines whether to generate (sample) a word from the vocabulary or to copy the word directly from the input text, based on attention distribution a_i^t:

$$p_{cg} = sigmoid(W_{eh}^T c_t^* + W_{sh}^T s_t + W_x x_t + b_{cg}) \quad (6)$$

Here W_{eh}, W_{sh}, W_x and b_{cg} are trainable model parameters. The final probability of decoding a word is specified by the mixture model:

$$p^*(qword) = p_{cg} \sum_{i:w_i=qword} a_i^t + (1-p_{cg})p(qword) \quad (7)$$

Where $p^*(qword)$ is the final distribution over the union of the vocabulary and the input sentence.

As discussed earlier, Equation (7) addresses the rare words issue, since a word not in vocabulary will have probability $p(qword) = 0$. Therefore, in such cases, our model will replace the $<unk>$token for out-of-vocabulary words with a word in the input sentence having the highest attention obtained using attention distribution a_i^t.

To discourage meaningless multiple repetitions of words in the question (as motivated in Section 1), we maintain a word coverage vector (wcv) for the words already predicted as the sum of all the attention distributions ranging over timesteps 0 until $t-1$. Specifically, wcv at time step t is:

$$wcv^t = \sum_{t'=0}^{t-1} a^{t'} \quad (8)$$

No word is generated before timestep 0, and hence *wcv* will be a zero vector then. After storing the word coverage vector until $t-1$, while attending to the next word, we will need to inform our attention mechanism about words covered until then. Hence, the update equation (3) is now modified to be:

$$e_i^t = v^t tanh(W_{wcv} wcv_i^t + W_{eh} h_i + W_{sh} s_t + b_{att}) \quad (9)$$

Here W_{wcv} are trainable parameters that inform the attention mechanism about words that have been previously covered while choosing to attend over the next word. Following the incorporation of the copy and coverage mechanism in our sequence-to-sequence architecture, the final loss function will be:

$$Loss_{copy+cov} = \frac{1}{T}\sum_{t=0}^{T} \log P^*(w_t) - \lambda_c L_{cov} \quad (10)$$

where λ_c is the coverage hyperparameter and the coverage loss L_{cov} is defined as:

$$L_{cov} = \sum_i min(a_i^t, wcv_i^t) \quad (11)$$

We note that this cross-entropy based loss function does not still include a task-specific metric such as BLEU, *etc.*, that was motivated earlier. We use RL to refine the model pre-trained on this loss function to directly optimize the task specific reward. We also empirically show that the refinement of maximum likelihood models using task-specific rewards such as BLEU improves results considerably. In the next subsection we describe our evaluator.

3.2 Evaluator

The evaluator helps in fine-tuning the parameters of the generator network. It takes as input the predicted sequence and the gold sequence, evaluates a policy and returns a reward (a score between 0 and 1) that reflects the quality of the question generated. For question generation, the choice of reward functions include task-specific metrics BLEU, GLEU and ROUGE-L (Du et al., 2017; Kumar et al., 2018). Our

framework also allows the flexibility to define custom reward functions, such as the decomposable attention described below.

3.2.1 Decomposable attention based evaluator

Use of a lexical similarity based reward function such as BLEU or ROUGE does not still provide the flexibility to handle multiple possible versions of the ground truth. For example, the questions *"who is the widow of ray croc?"* and *"ray croc was married to whom?"* have almost the same meaning, but due to word order mismatch with the gold question, at most one of them can be rewarded using the BLEU score at the cost of the other(s). Empirically, we find this restriction leading to models that often synthesize questions with poor quality.

We therefore design a novel reward function, a decomposable attention (Parikh et al., 2016) based similarity scorer (DAS). Denoting by $\hat{q}$ a generated question and by q a ground truth question, we compute a cross attention based similarity using the following steps:

Cross Attention: The generated question $\hat{q}$ and a ground truth question q are inter-attended as:

$$\hat{q}_i^* = \sum_{j=0}^{L_q} a_{ji} e(q_j),\ a_{ji} = \frac{\exp(e(\hat{q}_i)^T e(q_j))}{\sum_{k=0}^{L_{\hat{q}}} \exp(e(\hat{q}_i)^T e(q_k))},$$
$$q_j^* = \sum_{i=0}^{L_{\hat{q}}} b_{ji} e(\hat{q}_i),\ b_{ji} = \frac{\exp(e(\hat{q}_i)^T e(q_j))}{\sum_{k=0}^{L_q} \exp(e(\hat{q}_k)^T e(q_j))} \quad (12)$$

where $e(.)$ is the word embedding of dimension size d, $\hat{q}^*$ is the cross attention vector for a generated question $\hat{q}$, and q^* is the cross attention vector for a question q in the ground truth.

Compare: Each n-gram $\hat{q}_i$ in the generated question (through its embedding $e(\hat{q}_i)$) is compared with its associated cross-attention vector $\hat{q}^*$ using a feed forward neural network N_1. Similarly, each n-gram q_j in the ground truth question (through its embedding $e(q_j)$) is compared with its associated attention vector q^* using another network N_2 having the same architecture as N_1. The motivation for this comparison is that we would like to determine the soft alignment between n-grams in the generated question and the gold question. As an illustration, while comparing the gold question "*why do rockets look white?*" with a generated question "*why are rockets and boosters painted white?*", we find that an n-gram "*rockets and boosters*" is softly aligned to "*rockets*" while "*look*" is softly aligned to "*painted*".

$$\hat{\boldsymbol{q}}_{\boldsymbol{1},\boldsymbol{i}} = N_1([e(\hat{q}_i), \hat{q}^*]),\ \boldsymbol{q}_{\boldsymbol{2},\boldsymbol{j}} = N_2([e(q_j), q^*]) \quad (13)$$

where $\hat{\boldsymbol{q}}_{\boldsymbol{1},\boldsymbol{i}}$ and $\boldsymbol{q}_{\boldsymbol{2},\boldsymbol{j}}$ are vectors containing comparison scores of aligned phrases in generated question and gold question respectively and N_1 and N_2 are the feed forward neural nets.

Matching Score: The vectors $\hat{\boldsymbol{q}}_{\boldsymbol{1},\boldsymbol{i}}$ and $\boldsymbol{q}_{\boldsymbol{2},\boldsymbol{j}}$ are aggregated over each word or phrase in the predicted question and gold question respectively before feeding them to a linear function (L):

$$DAS = L(\sum_{i=1}^{L_q} \hat{\boldsymbol{q}}_{\boldsymbol{1},\boldsymbol{i}}, \sum_{j=1}^{L_{\hat{q}}} \boldsymbol{q}_{\boldsymbol{2},\boldsymbol{j}}) \quad (14)$$

This matching score between the predicted question and the gold question is the reward associated with the decomposable attention based decoder.

3.3 Overall Loss Function

Combining Equation (10) with a reward function R (BLEU, GLEU, ROUGE or the decomposable attention based function DAS in Equation (14)), we obtain the overall loss function using expected reward objective as follows:

$$L_{overall} = \alpha * Loss_{copy+cov} + \beta * \sum_{i=0}^{N} \sum_{y \in \mathcal{Y}} P_\theta(y|X^{(i)}) R(y, y^{*(i)}) \quad (15)$$

where $R(y, y^{*(i)})$ denotes per sentence score (reward), $\mathcal{Y}$ is a set of sequences sampled from the final distribution, and α and β are tunable hyperparamters.

4 Experimental Setup

In this section, we present our experimental results on the processed publicly available SQuAD (Rajpurkar et al., 2016) dataset. We first describe our dataset for the task. Next we explain various reward functions employed in our experiments. We then describe implementation details of our framework and finally present our evaluation methods.

4.1 Dataset

The benchmark dataset SQuAD (Rajpurkar et al., 2016) was used in all our experiments. SQuAD contains sentences (32.9 tokens on average) paired with questions (1.4 questions per sentence with 11.3 tokens on average). The dataset has 70,484 training

pairs, 10,570 validation pairs and 11,877 test pairs. We augmented each sentence with a rich set of linguistic features such as POS tags, named entity tags, and dependency labels.

4.2 Reward Functions

We experimented with four reward functions (1) BLEU, (2) GLEU, (3) ROUGE-L, and (4) DAS (decomposable attention-based similarity score, explained in Section 3.2.1). Of these, BLEU and ROUGE-L are standard metrics used at test time to evaluate question generation systems. In our experiments we considered BLEU for up to 4-grams. For the GLEU score, we recorded all sub-sequences of up to 4-grams.

4.3 Implementation Details

We implemented our framework[2] in Python using Tensorflow version 1.4. We used a two-layer bidirectional LSTM for the encoder and a single-layer LSTM for the decoder in the generator model. We set the number of LSTM hidden states to 256. The vocabulary size was fixed to 60,000 words for both the sentence and the question. We initialized the word embeddings using pre-trained 300-dimensional GLOVE.840B.300D, which was fixed during training.

Optimization is performed using AdaGrad with a learning rate 0.15 during pre-training with cross-entropy loss and with learning rate 0.01 during fine-tuning with the policy gradient algorithm. We fix the batch size for training to be 16. We train the model with cross-entropy loss for 30 epochs. We then fine-tune the generator using our policy gradient algorithm for further 15 epochs. We set the coverage hyperparameter λ to 0.5 and train for a further 4,505 iterations (approx one epoch). We found it ineffective to training with coverage right from the beginning. During reinforcement training using the policy gradient algorithm the best performance of the model was achieved by setting the hyperparameters α and β in (15) to 0.015 and 1 respectively. During decoding we perform beam search with beam size of 3.

We train our decomposable attention-based similarity scorers (N_1 and N_2 in Formula 13) using publicly available Quora question pairs dataset[3] for 50 epochs and use this trained model to calculate the similarity score between the sampled question and the gold question. We tune our hyperparameters on the development set and report evaluation results on the test set.

[2]The code and data will be released upon publication (code and data is attached as supplementary material).

[3]https://goo.gl/k5UjpT

4.4 Evaluation Methods

We implement two state-of-the-art question generation models: L2A (Du et al., 2017) and AutoQG (Kumar et al., 2018) as baselines for comparison. We report automatic and human evaluation results on our four RL models, each of which is equipped with the copy and coverage mechanism as well as one of the four reward functions: BLEU, GLEU, ROUGE-L, and DAS. Hence, our models are named RL_{BLEU}, etc.

For automatic evaluation we use standard evaluation measures used to evaluate sequence prediction tasks. We use BLEU, ROUGE-L and METEOR evaluation scripts released by Chen et al (2015), which was originally used to evaluate the image captioning task.

We also performed human evaluation to further analyze the quality of questions generated for their syntactic correctness, semantic correctness and relevance. Syntactic correctness measures the grammatical correctness of a generated question, semantic correctness measures meaningfulness and naturalness of the question, and relevance measures how relevant the question is to the text. We perform human evaluation for each model on a randomly selected subset of 100 sentences. Each of the three judges is presented the 100 sentence-question pairs for each model and asked for a binary response on each quality parameter. The responses from all the judges for each parameter is then averaged for each model.

5 Results and Discussion

We show and compare results with automatic evaluation metrics in Table 2. As can be seen, all our four models outperform the two baseline models on all evaluation metrics. Hence, using any evaluation metric as the reward function during reinforcement based learning improves performance for all metrics. We also observe that RL_{ROUGE}, the model reinforced with ROUGE-L (that measures the longest common sequence between the ground-truth question and the generated question) as the reward function, is the best performing model on all metrics, outperforming existing baselines considerably. For example, it improves over AutoQG on BLEU-4 by 27.5%, on METEOR by 11%, and on ROUGE-L by 8%.

5.1 Analyzing Choice of Reward Function

BLEU measures precision: how many words (and/or n-grams) in machine generated questions appear in the ground-truth questions. Whereas ROUGE measures recall: how many words (and/or n-grams)

in the ground-truth questions appear in the machine generated questions. We believe that cross-entropy loss was already accounting for precision to some extent and using it in conjunction with ROUGE (which improves recall) therefore gives best performance.

In Table 3 we present human evaluation results for the models evaluated on three quality parameters (a) syntactic correctness, (b) semantic correctness, and (c) relevance.

Consistent with automatic evaluation results shown in Table 2, three of our four models outperform the two baselines, with RL_{ROUGE} again being the best model on all three quality metrics, outperforming all the other models by a large margin. We can also observe that, compared to the two baselines and RL_{BLEU}, RL_{DAS} significantly improves quality of questions generated even though there is less noticeable improvements in the BLEU scores.

We measure inter-rater agreement using Randolph's free-marginal multirater kappa (Randolph, 2005). This helps in analyzing level of consistency among observational responses provided by multiple judges. It can be observed that our quality metrics for all our models are rated as *moderate agreement* (Viera et al., 2005).

In Table 5 we present one example sentence, the ground-truth question, and the questions generated by each model. We can observe that RL_{ROUGE} generates syntactically and semantically correct and relevant question.

DAS calculates semantic similarity between generated question and the gound-truth question. For example if the gound-truth question is: "who is the widow of mcdonald's owner?" And the generated question is "mcdonald's owner is married to whom ?" or "who is married to mcdonald's owner?" DAS will give high reward even though the generated question has low BLEU score. Thus, the performance of the model on automatic evaluation metrics does not improve with DAS as the reward function, though the quality of questions certainly improves. Further, ROUGE in conjunction with the cross entropy loss improves on recall as well as precision whereas every other combination overly focuses only on precision.

Error analysis of our best model reveals that most errors can be attributed to intra sentence dependencies such as co-references, concept dependencies *etc.* In Section 3 of the supplementary material, we have presented examples on each of which RL_{ROUGE} and every other model is unable to generate a syntactically correct question that consequently becomes less meaningful and irrelevant in most cases.

5.2 Ablation Analysis

We have conducted an ablation analysis to study the effect of removing the copy and coverage mechanisms. Table 4 summarizes the drop in performance for RL_{ROUGE}. Without the copy mechanism, there is a drop overall in every evaluation measure, with BLEU-4 registering the largest drop of 13.8% as against 13.4%, 6.9% and 4.7% in BLEU-3, BLEU-2 and BLEU-1 respectively. On the other hand, without the coverage mechanism, we see a consistent but sufficiently lower drop (1-2%) in each evaluation measure for RL_{ROUGE}. Despite this seemingly small drop, we observe that the *quality* of questions generated using our complete model with the coverage mechanism is significantly better than the quality without it. The results for RL_{ROUGE} without both copy and coverage mechanisms are only slightly worse than without the copy mechanism alone.

6 Related Work

Existing question generation approaches can be broadly classified into four categories: (a) rule-based, (b) template based, (c) semantics based, and (d) more recently, neural network based. Heilman (2011) presents a rule-based system that first simplifies complex inputs into simplified factual statements by modifying syntactic structures and transforming lexical items. Questions are generated from these simplified sentences using simple, linguistic transformations such as WH-movement and subject-auxiliary inversion. Linguistic properties of input sentences are analyzed using a pipeline of NLP tools.

The basis for template-based approaches (Mostow and Chen, 2009; Hussein et al., 2014) is that a question template captures the class of context-specific questions having a common structure. Some examples of question templates are "*What is the capital of* <X>*?*" and "*What would happen if* <Y>*?*", where<X>and<Y>are placeholders. These template based and rule based approaches rely heavily on limited set of handcrafted rules and templates.

Motivated by neural machine translation, Du et al. (2017) recently proposed a sequence-to-sequence architecture for automatic question generation from text. Serban et al. (2016) proposed a recurrent neural network (RNN) based architecture to automatically generate question from Freebase triples, each of which comprises a subject, a predicate and an object. Kumar et al. (2018) proposed to embed each word with rich

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L
L2A	43.21	24.77	15.93	10.60	16.39	38.98
AutoQG	44.68	26.96	18.18	12.68	17.86	40.59
RL_{BLEU}	46.84	29.38	20.33	14.47	19.08	41.07
RL_{DAS}	44.64	28.25	19.63	14.07	18.12	42.07
RL_{GLEU}	45.20	29.22	20.79	15.26	18.98	43.47
RL_{ROUGE}	**47.01**	**30.67**	**21.95**	**16.17**	**19.85**	**43.90**

Table 2: Experimental results on the test set on automatic evaluation metrics. Best results for each metric (column) are **bolded**.

Model	Syntactically correct		Semantically correct		Relevant	
	Score	Kappa	Score	Kappa	Score	Kappa
L2A	39.2	0.49	39	0.49	29	0.40
AutoQG	51.5	0.49	48	0.78	48	0.50
RL_{BLEU}	47.5	0.52	49	0.45	41.5	0.44
RL_{DAS}	68	0.40	63	0.33	41	0.40
RL_{GLEU}	60.5	0.50	62	0.52	44	0.41
RL_{ROUGE}	**69.5**	0.56	**68**	0.58	**53**	0.43

Table 3: Human evaluation results (column "Score") as well as inter-rater agreement (column "Kappa") for each model on the test set. The scores are between 0-100, 0 being the worst and 100 being the best. Best results for each metric (column) are **bolded**.

Model (RL_{ROUGE})	**Δ BLEU-1 (47.01)**	**Δ BLEU-2 (30.67)**	**Δ BLEU-3 (21.95)**	**Δ BLEU-4 (16.17)**	**Δ METEOR (19.85)**	**Δ ROUGE-L (43.90)**
W/o copy	2.09 (4.7%)	2.13 (6.9%)	2.94 (13.4%)	2.23 (13.8%)	2.21 (11.1%)	2.58 (5.9%)
W/o coverage	0.31 (0.7%)	0.57 (1.9%)	0.94 (4.2%)	0.28 (1.7%)	0.84 (4.2%)	1.01 (2.3%)

Table 4: Ablation analysis results after removing (a) copy mechanism and (b) coverage mechanism from the best performing system (RL_{ROUGE}). Both absolute performance drop and percentage of drop (in parenthese) are reported.

Text: "critics such as economist paul krugman and u.s. treasury secretary timothy geithner have argued that the regulatory framework did not keep pace with financial innovation , such as the increasing importance of the shadow banking system , derivatives and off-balance sheet financing ."
Ground-truth: it has been argued that what did not keep up with financial innovation ?

Model	**Question**
L2A	what was the u.s. of the executive secretary argued that the framework framework did keep be pace ?
AutoQG	who argued that the regulatory framework was not keep to take pace with financial innovation ?
RL_{BLEU}	what was the name of the increasing importance of the shadow banking system ?
RL_{DAS}	what was the main focus of the problem with the shadow banking system ?
RL_{GLEU}	what was not keep pace with financial innovation ?
RL_{ROUGE}	what did paul krugman and u.s. treasury secretary disagree with ?

Table 5: An example sentence and ground-truth question in SQuAD as well as the questions generated by the six models.

set of linguistic features and encoded most relevant pivotal answer to the text while generating question .

Almost all neural network based techniques optimize cross-entropy loss. However, log likelihood/cross-entropy loss optimization based methods are limited, since they learn to predict the next token based on the previous prediction, and are not trained on the ground-truth question in its entirety. (Ranzato et al., 2015) incrementally trains for the first s steps on the cross entropy loss then for the subsequent t-s steps, it trains using REINFORCE (Sutton and Barto, 1998) and this process repeats. In contrast, we jointly optimize both cross entropy and reinforcement losses with a tradeoff.

7 Conclusion

In this work we present a novel approach to automatically generating questions from text using deep reinforcement learning. A copy and coverage mechanism to handle rare word and repetition problems is also incorporated. Our framework allows us to directly optimize any task-specific score including evaluation measures such as BLEU, GLEU, ROUGE-L, and decomposable attention that are naturally suited to QG and other seq2seq problems. Experimental results on automatic evaluation and human evaluation on the standard benchmark dataset show that our proposed approach considerably outperforms state-of-the-art systems.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft COCO captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Colleen E Crangle and Joyce Brothers Kart. 2015. A questions-based investigation of consumer mental-health information. *PeerJ*, 3:e867.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1342–1352.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1631–1640.

Michael Heilman. 2011. *Automatic factual question generation from text*. Ph.D. thesis, Carnegie Mellon University.

Hafedh Hussein, Mohammed Elmogy, and Shawkat Guirguis. 2014. Automatic english question generation system based on template driven scheme. *International Journal of Computer Science Issues (IJCSI)*, 11(6):45.

Vishwajeet Kumar, Kireeti Boorla, Yogesh Meena, Ganesh Ramakrishnan, and Yuan-Fang Li. 2018. Automating reading comprehension by generating question and answer pairs. In *22nd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Prashanth Mannem, Rashmi Prasad, and Aravind Joshi. 2010. Question generation from paragraphs at UPenn: QGSTEC system description. In *Proceedings of QG2010: The Third Workshop on Question Generation*, pages 84–91.

Jack Mostow and Wei Chen. 2009. Generating instruction automatically for the reading strategy of self-questioning. In *AIED*, pages 465–472.

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*, pages 311–318. Association for Computational Linguistics.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255.

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, Austin, Texas. Association for Computational Linguistics.

Justus J Randolph. 2005. Free-marginal multirater kappa (multirater k [free]): An alternative to fleiss' fixed-marginal multirater kappa. *Online submission*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. *arXiv preprint arXiv:1603.06807*.

Richard S Sutton and Andrew G Barto. 1998. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, pages 76–85. The Association for Computer Linguistics.

Anthony J Viera, Joanne M Garrett, et al. 2005. Understanding interobserver agreement: the kappa statistic. *Fam Med*, 37(5):360–363.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.