Generating Distractors for Reading Comprehension Questions from Real Examinations

Yifan Gao,[1]* Lidong Bing,[2]† Piji Li,[3] Irwin King,[1] Michael R. Lyu[1]
[1]Department of Computer Science and Engineering,
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
[2]R&D Center Singapore, Machine Intelligence Technology, Alibaba DAMO Academy
[3]Tencent AI Lab
[1]{yfgao,king,lyu}@cse.cuhk.edu.hk [2]l.bing@alibaba-inc.com [3]pijili@tencent.com

Abstract

We investigate the task of distractor generation for multiple choice reading comprehension questions from examinations. In contrast to all previous works, we do not aim at preparing words or short phrases distractors, instead, we endeavor to generate longer and semantic-rich distractors which are closer to distractors in real reading comprehension from examinations. Taking a reading comprehension article, a pair of question and its correct option as input, our goal is to generate several distractors which are somehow related to the answer, consistent with the semantic context of the question and have some trace in the article. We propose a hierarchical encoder-decoder framework with static and dynamic attention mechanisms to tackle this task. Specifically, the dynamic attention can combine sentence-level and word-level attention varying at each recurrent time step to generate a more readable sequence. The static attention is to modulate the dynamic attention not to focus on question irrelevant sentences or sentences which contribute to the correct option. Our proposed framework outperforms several strong baselines on the first prepared distractor generation dataset of real reading comprehension questions. For human evaluation, compared with those distractors generated by baselines, our generated distractors are more functional to confuse the annotators.

Introduction

Reading comprehension (RC) is regarded as an avant-garde task in NLP research for practising the capability of language understanding. Models with recent advances of deep learning techniques are even capable of exceeding human performance in some RC tasks, such as for questions with span-based answers (Yu et al. 2018). However, it is not the case when directly applying the state-of-the-art models to multiple choice questions (MCQs) in RACE dataset (Lai et al. 2017), elaborately designed by human experts for real examinations, where the task is to select the correct answer from a few given options after reading the article. The performance gap between the state-of-the-art deep models (53.3%) (Tay, Tuan, and Hui 2018) and ceiling (95%) (Lai et al. 2017) is significant. One possible reason is that in MCQs, besides the *question* and the correct *answer* option, there are a few *distractors* (wrong options) to distract humans or machines from the correct answer. Most distractors are somehow related to the answer and consistent with the semantic context of the question, and all of them have correct grammar (Goodrich 1977; Liang et al. 2018; Ma, Lyu, and King 2010). Furthermore, most of the distractors have some trace in the article, which fails the state-of-the-art models utilizing context matching only to yield decent results.

Article:
. . .
The Yanomami live along the rivers of the rainforest in the north of Brazil. They have lived in the rainforest for about 10,000 years and they use more than 2,000 different plants for food and for medicine. But in 1988, someone found gold in their forest, and suddenly 45,000 people came to the forest and began looking for gold. They cut down the forest to make roads. They made more than a hundred airports. The Yanomami people lost land and food. Many died because new diseases came to the forest with the strangers.
. . .
In 1987, they closed fifteen roads for eight months. No one cut down any trees during that time. In Panama, the Kuna people saved their forest. They made a forest park which tourists pay to visit. The Gavioes people of Brazil use the forest, but they protect it as well. They find and sell the Brazil nuts which grow on the forest trees.

Question:
Those people built roads and airports in order to _ .

A. carry away the gold conveniently (**Answer**)
B. make people there live a better life (**Distractor**)
C. stop spreading the new diseases (**Distractor**)
D. develop the tourism there (**Distractor**)

Figure 1: Sample multiple choice question along with the corresponding article. The question, options and their relevant sentences in the article are marked with the same color.

The MCQs in the RACE dataset are collected from the English exams for Chinese students from grade 7 to 12. Constructing RACE-like MCQ dataset is important and nontriv-

*This work was mainly done when Yifan Gao was an intern at Tencent AI Lab.

†This work was mainly done when Lidong Bing was working at Tencent AI Lab.

ial, because poor distractor options can make the questions almost trivial to solve (Welbl, Liu, and Gardner 2017) and reasonable distractors are time-consuming to design. In this paper, we investigate the task of automatic *distractor generation* (DG) . The task aims to generate reasonable distractors for RACE-like MCQs, given a reading comprehension article, and a pair of question and its correct answer originated from the article. Figure 1 shows an example multiple choice question with four options. We can find that all options are grammatically coherent with the question, and semantically relevant to the article. Distractor generation is of great significance in a few aspects. It can aid the preparation of MCQ reading comprehension datasets. With large datasets prepared, it is expectable that the performance of reading comprehension systems for MCQs will be boosted, as we have observed such improvements (Yang et al. 2017) by applying generated question-answer pairs to train models to solve SQuAD questions. It could also be helpful to alleviate instructors' workload in designing MCQs for students.

Automatic DG is different from previous distractor preparation works, which basically follow an extraction-selection manner. First, a distractor candidate set is extracted from multiple sources, such as GloVe vocabulary (Pennington, Socher, and Manning 2014), noun phrases from textbooks (Welbl, Liu, and Gardner 2017) and articles (Araki et al. 2016). Then similarity based (Guo et al. 2016; Stasaski and Hearst 2017; Kumar, Banchs, and D'Haro 2015; Mitkov 2003; Zhou, Lyu, and King 2012; Yang et al. 2011) or learning based (Liang et al. 2018; Sakaguchi, Arase, and Komachi 2013; Liang et al. 2017; Yang et al. 2010) algorithms are employed to select the distractors. Another manner is to apply some pre-defined rules to prepare distractors by changing the surface form of some words or phrases (Chen, Liou, and Chang 2006). Automatic DG for RACE-like MCQs is a challenging task. First, different from previous works that prepare word or short phrase distractors (1.46 tokens on average in SciQ (Welbl, Liu, and Gardner 2017)), we here endeavor to generate longer and semantic-rich distractors. Specifically, the average length of the distractors in our experimental dataset is 8.1. Furthermore, the generated distractors should semantically related to the reading comprehension question, since it is trivial to identify a distractor having no connection with the article or question. Moreover, the distractors should not be paraphrases of the correct answer option. Finally, the generated distractors should be grammatically consistent with the question, especially for questions with a blank in the end, as shown in Figure 1. Previous works following the extraction-selection manner cannot meet these requirements.

We formulate the task of automatic distractor generation as a sequence-to-sequence learning problem that directly generates the distractors given the article, and a pair of question and its correct answer. We design our framework to explicitly tackle the above mentioned challenges by using a data-driven approach to learn to meet these requirements automatically. More specifically, we employ the hierarchical encoder-decoder network, which has already shown potentials to tackle long sequential input (Tan, Wan, and Xiao 2017; Ling and Rush 2017), as the base model for building our framework. On top of the hierarchical encoding structure, we propose the dynamic attention mechanism to combine sentence-level and word-level attentions varying at each recurrent time step to generate a more readable sequence. Furthermore, a static attention mechanism is designed to modulate the dynamic attention not to focus on question-irrelevant sentences or sentences which contribute to the correct answer option. Finally, we use a question-based initializer as the start point to generate the distractor, which makes the distractor grammatically consistent with the question. In the generation stage, we use the beam search to generate three diverse distractors by controlling their distance.

In the evaluations, we conduct experiments on a distractor generation dataset prepared from RACE using n-gram based automatic evaluation metrics such as BLEU and ROUGE. The results show that our proposed model beats several baselines and ablations. Human evaluations show that distractors generated by our model are more likely to confuse the examinees, which demonstrates the functionality of our generated distractors in real examinations. We will release the prepared dataset and the code of our model to facilitate other researchers to do further research along this line [1].

Framework Description

Task Definition

In the task of automatic Distractor Generation (DG), given an article, a pair of question and its correct option originated from the article, our goal is to generate context and question related, grammatically consistent wrong options, i.e. distractor, for the question.

Formally, let P denote the input article containing multiple sentences: $s_1, s_2, ..., s_n$, q and a denote the question and its correct answer, respectively. The DG task is defined as finding the distractor $\overline{d}$, such that:

$$\overline{d} = \arg\max_{d} \log \mathrm{P}(d|P, a, q), \tag{1}$$

where $\log \mathrm{P}(d|P, a, q)$ is the conditional log-likelihood of the predicted distractor d, give P, a and q.

Framework Overview

A straightforward strategy for distractor generation is to employ the standard sequence-to-sequence learning network (Sutskever, Vinyals, and Le 2014) to learn the mapping from the article to the distractor. Unfortunately, an article can be too long as the input, which cannot receive decent results. Here we advocate the hierarchical encoder-decoder framework to model such long sequential input. The architecture of our overall framework is depicted in Figure 2.

First, we employ the hierarchical encoder to obtain hierarchical contextualized representations for the whole article, namely, word-level representation and sentence-level representation. Before decoding the encoded information, we design a static attention mechanism to model the global sentence importance considering the fact that the distractor

[1] Our code and data are available at https://github.com/Evan-Gao/Distractor-Generation-RACE

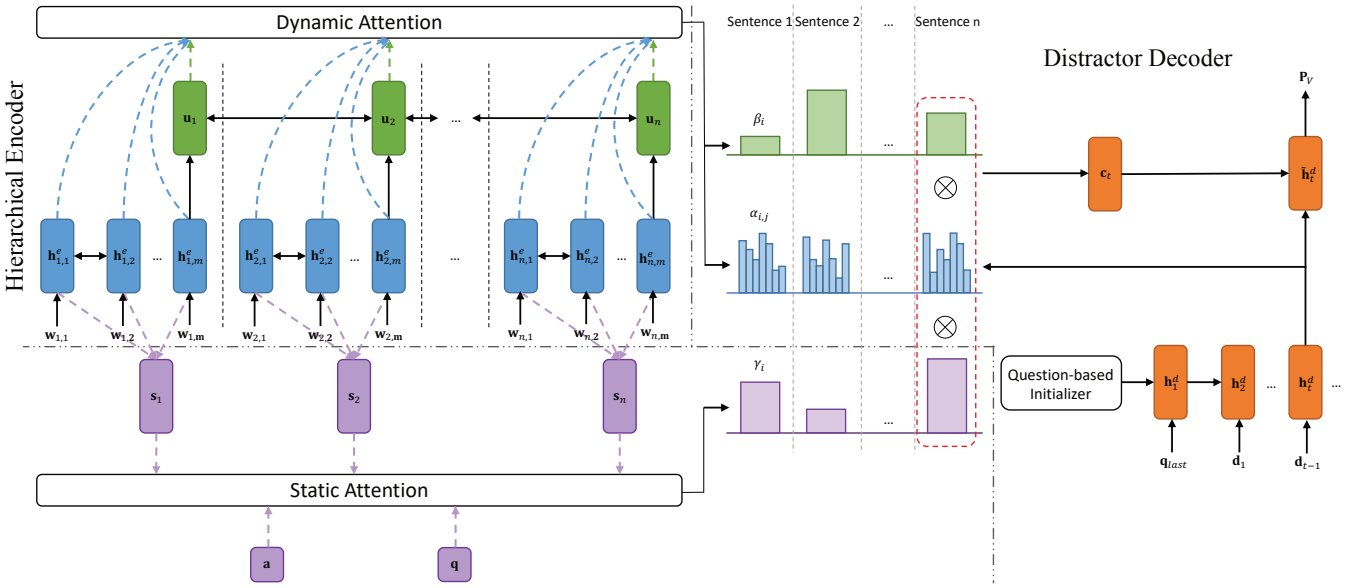

Figure 2: A overview of our model that jointly utilizes static and dynamic attentions. *(Better viewed in color).*

should be semantically related to the question and should not share the same semantic meaning with the correct answer. The static attention distribution is used in the decoder as a soft gate to modulate the dynamic attention. For the decoder part, we first employ a language model to compress the question information into a fixed length vector to initialize the decoder state, making the distractor grammatically consistent with the question. During each decoding step, the dynamic hierarchical attention combines the sentence-level and word-level information to attend different part at each decoding time step. With the combined architecture, our model can generate grammatically consistent, context and question related wrong options (distractors) in an end-to-end manner.

Hierarchical Encoder

Word Embedding. An embedding lookup table is firstly used to map tokens in each sentence s_i in the article P into word dense vectors $(\mathbf{w}_{i,1}, \mathbf{w}_{i,2}, ..., \mathbf{w}_{i,m})$, where $\mathbf{w}_{i,j} \in \mathbb{R}^{d_w}$ having d_w dimensions.

Word Encoder. For each sentence s_i, the word encoder takes its word vectors $(\mathbf{w}_{i,1}, \mathbf{w}_{i,2}, ..., \mathbf{w}_{i,m})$ as input. Specifically, we use bidirectional LSTMs to encode the sequence to get a contextualized representation for each word:

$$\overrightarrow{\mathbf{h}^e_{i,j}} = \overrightarrow{\text{LSTM}}(\overrightarrow{\mathbf{h}^e_{i,j-1}}, \mathbf{w}_{i,j}), \ \overleftarrow{\mathbf{h}^e_{i,j}} = \overleftarrow{\text{LSTM}}(\overleftarrow{\mathbf{h}^e_{i,j+1}}, \mathbf{w}_{i,j}),$$

where $\overrightarrow{\mathbf{h}^e_{i,j}}$ and $\overleftarrow{\mathbf{h}^e_{i,j}}$ are the hidden states at the j-th time step of the forward and the backward LSTMs. We concatenate them together as $\mathbf{h}^e_{i,j} = [\overrightarrow{\mathbf{h}^e_{i,j}}; \overleftarrow{\mathbf{h}^e_{i,j}}]$.

Sentence Encoder. On top of the word encoding layer, we combine the final hidden state of the forward LSTM and the first hidden state of the backward LSTM of each sentence as the sentence representation and employ another bidirectional LSTMs to learn the contextual connection of sentences. We denote the contextualized representation of the sentence sequence as $(\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n)$.

Static Attention Mechanism

Recall that the generated distractors should be semantically relevant to the question, but must not share the same semantic meaning with the answer. To achieve this goal, here we introduce a static attention mechanism which learns an importance distribution $(\gamma_1, \gamma_2, ..., \gamma_n)$ of the sentences $(s_1, s_2, ..., s_n)$ in the article. Here we use the answer a and the question q as queries to interact with all sentences to learn such distribution.

Encoding Layer. In the encoding layer, we transform the answer a, the question q and all sentences $(s_1, s_2, ..., s_n)$ into fixed length vector representations. Specifically, two individual bidirectional LSTM networks are employed to encode a and q separately to derive the contextualized representation for each token in them and obtain $(\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_k)$ and $(\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_l)$, respectively. Then an average pooling layer is employed to acquire the representation for the question and answer:

$$\mathbf{a} = \frac{1}{k}\sum_{t=1}^{k}\mathbf{a}_t, \mathbf{q} = \frac{1}{l}\sum_{t=1}^{l}\mathbf{q}_t. \quad (2)$$

For the sentence representation, we do not reuse the sentence representation $\mathbf{u}_i$ from the sentence encoder since $\mathbf{u}_i$ is responsible for learning the semantic information for a whole sentence, while here we only want to learn the importance distribution of sentences according to the query (i.e. a pair of question and answer). Therefore, we only reuse the word-level contextualized representations $\mathbf{h}^e_{i,j}$ learned in the hierarchical encoder and employ the same average pooling layer

to get the representation of each sentence:

$$\mathbf{s}_i = \frac{1}{m}\sum_{t=1}^{m}\mathbf{h}^e_{i,t}. \quad (3)$$

Matching Layer. For generating non-trivial distractors, we should emphasize the sentences that are relevant to the question, and suppress the sentences relevant to the answer. For this reason, we learn a score o_i for s_i that combines the above two aspects with bilinear transformation similar to (Chen et al. 2018):

$$o_i = \lambda_q \mathbf{s}_i^\top \mathbf{W}_m \mathbf{q} - \lambda_a \mathbf{s}_i^\top \mathbf{W}_m \mathbf{a} + \mathbf{b}_m, \quad (4)$$

where $\mathbf{W}_m$ and $\mathbf{b}_m$ are learnable parameters.

Normalization Layer. Before feeding the raw sentence importance score o_i into the Softmax function to compute the final static attention distribution, we use the question to learn a temperature $\tau \in (0, 1)$:

$$\tau = \text{sigmoid}(\mathbf{w_q}^\top \mathbf{q} + b_q), \quad (5)$$

where $\mathbf{w_q}$ and b_q are learnable parameters. Then, we derive the static attention distribution as:

$$\gamma_i = \text{softmax}(o_i/\tau). \quad (6)$$

The intuition behind using the temperature τ is that if a question asks for some specific details in the article, it is only relevant to one or two sentences. While if a question requires summarizing or reasoning, it could be relevant to many sentences in the article. Therefore, we propose the above data-driven approach to learn the temperature τ according to the property of the question. If τ is close to 0, then it works together with o_i to yield a peaked distribution γ which simulates the case of detailed questions. Otherwise, if τ is close to 1, it will not peak any sentence attention score γ_i.

Distractor Decoder

We use another LSTMs as the decoder to generate the distractor. Instead of using the last hidden state of the encoder to initialize the decoder, we design a special question-based initializer to make the distractor grammatically consistent with the question. During the decoding, we introduce the dynamic attention mechanisms to combine the sentence-level and word-level attentions varying at each recurrent time step to generate a more readable sequence. We also incorporate the static attention here to modulate the dynamic attention to ensure the semantic relevance of the generated distractors.

Question-based Initializer. We design a question-based initializer to initialize the initial state of the decoder. Specifically, we use a question LSTM to encode the question, and use the last time step information of the LSTM in the following manner:

- Instead of using BOS (i.e. the *Begin of Sentence* indicator), we use the last token in the question ($\mathbf{q}_{last}$) as the initial input of the decoder.
- Other than using the final state of the hierarchical encoder to initialize the decoder, we here use the final cell state and hidden state of the question LSTM to initialize the decoder.

Dynamic Hierarchical Attention Mechanism. The standard attention mechanism treats an article as a long sequence and compares the hidden state of the current decoding time step to all encoder hidden states. This approach is not suitable for long input sequences for the following reasons. First, the standard LSTM cannot model such long inputs (on average, 343.9 words per article in our training set). Second, we will lose the sentence structure if we treat the tokens of different sentences equally. Last but not least, usually a question or a distractor is only related to a small number of sentences in the article, we should only use the related sentences to generate the distractor, but the standard attention has no emphasis on difference sentences.

Given the above reasons, we employ the dynamic hierarchical attention to only focus on important sentences during each decoding time step. We call it *dynamic* because both word-level and sentence-level attention distributions change at each time step. When generating a word at the time step t, the decoder reads the word embedding $\mathbf{d}_{t-1}$ and the hidden state $\mathbf{h}^d_{t-1}$ of the previous time step to generate the current hidden state $\mathbf{h}^d_t = \text{LSTM}(\mathbf{h}^d_{t-1}, \mathbf{d}_{t-1})$. Then it calculates both the sentence-level attention β_i and the word-level attention $\alpha_{i,j}$ at the same time:

$$\beta_i = \mathbf{u}_i^\top \mathbf{W}_{d_1} \mathbf{h}^d_t, \quad \alpha_{i,j} = \mathbf{h}^e_{i,j}{}^\top \mathbf{W}_{d_2} \mathbf{h}^d_t, \quad (7)$$

where $\mathbf{W}_{d_1}$ and $\mathbf{W}_{d_2}$ are trainable parameters. The sentence-level attention determines how much each sentence should contribute to the generation at the current time step, while the word-level attention determines how to distribute the attention over words in each sentence.

Finally, we use the static attention γ_i to modulate the dynamic hierarchical attention β_i and $\alpha_{i,j}$ by simple scalar multiplication and renormalization. Thus, the combined attention for each token in the article is:

$$\widetilde{\alpha}_{i,j} = \frac{\alpha_{i,j}\beta_i\gamma_i}{\sum_{i,j}\alpha_{i,j}\beta_i\gamma_i}. \quad (8)$$

Then the context vector $\mathbf{c}_t$ is derived as a combination of all article token representations reweighted by the final combined attention $\widetilde{\alpha}_{i,j}$:

$$\mathbf{c}_t = \sum_{i,j}\widetilde{\alpha}_{i,j}\mathbf{h}^e_{i,j}. \quad (9)$$

And the attentional vector is calculated as:

$$\tilde{\mathbf{h}}^d_t = \tanh(\mathbf{W}_{\tilde{\mathbf{h}}}[\mathbf{h}^d_t; \mathbf{c}_t]). \quad (10)$$

Then, the predicted probability distribution over the vocabulary V at the current step is computed as:

$$\text{P}_V = \text{softmax}(\mathbf{W}_V \tilde{\mathbf{h}}^d_t + \mathbf{b}_V), \quad (11)$$

where $\mathbf{W}_{\tilde{\mathbf{h}}}$, $\mathbf{W}_V$ and $\mathbf{b}_V$ are learnable parameters.

Training and Inference

Given the training corpus $\mathcal{Q}$ in which each data sample contains a distractor d, an article P, a question q and an answer a, we minimize the negative log-likelihood with respect to all learnable parameters Θ for training:

$$\mathcal{L} = -\sum_{d\in\mathcal{Q}} \log \text{P}(d|P, a, q; \Theta). \quad (12)$$

# Train Samples	96501
# Dev Samples	12089
# Test Samples	12284
Avg. article length (tokens)	347.0
Avg. distractor length	8.5
Avg. question length	9.9
Avg. answer length	8.7
Avg. # distractors per question	2.1

Table 1: The statistics of our dataset.

During generation, if UNK (i.e. unknown words) is decoded at any time step, we replace it with the word having the largest attention weight in the article.

Since there are several diverse distractors (2.4 on average according to Table 1) corresponding to the same question in our dataset, we use beam search with beam size k in the testing stage and receive k candidate distractors with decreasing likelihood. The ultimate goal is to generate several diverse distractors, however, usually the successive output sequences from beam search would be similar. Therefore we design the following protocol to generate three diverse distractors. Firstly, we select the distractor with the maximum likelihood as d_1^g. Then we select d_2^g among the remaining candidate distractors along the decreasing order of the likelihood, restricting that the Jaccard distance between d_1^g and d_2^g is larger than 0.5. Finally, d_3^g is selected in a similar way where its distances to both of d_1^g and d_2^g are restricted.

Experimental Settings

Dataset

We evaluate our framework on a distractor generation dataset prepared with the RACE (Lai et al. 2017) dataset. RACE contains 27,933 articles with 97,687 questions from English examinations of Chinese students from grade 7 to 12. We first extract each data sample as a quadruple of article, question, answer and distractor from RACE, followed by some simple preprocessing steps, such as tokenization, sentence splitting, and lower-casing.

After some investigation on the RACE dataset, we observe that some distractors have no semantic relevance with the article, which can be easily excluded in the examination and also do not make sense for the task of distractor generation since our goal is to generate confusing distractors. Hence, we first filter out such irrelevant distractors by simply counting meaningful tokens in individual distractors. We define a token meaningful if it is not a stop word and has a POS tag from {'JJ', 'JJR', 'JJS', 'NN', 'NNP', 'NNPS', 'NNS', 'RB', 'RBR', 'RBS', 'VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ'}. Then, we prune the dataset based on the following constraint: For those meaningful tokens in a distractor that also appear in the article, if their total weighted frequency is no less than 5, the distractor will be kept. Here the weighted frequency of a meaningful token means the multiplication of its frequency in the distractor and its frequency in the article. Moreover, we remove the questions which need to fill in the options at the beginning or in the middle of the questions. Table 1 reports the statistics of the processed dataset.

We randomly divide the dataset into the training (80%), validation (10%) and testing sets (10%).

Implementation Details

We keep the most frequent 50k tokens in the entire training corpus as the vocabulary, and use the `GloVe.840B.300d` word embeddings (Pennington, Socher, and Manning 2014) for initialization and finetune them in the training. Both source and target sides of our model share the same word embedding. All other tokens outside the vocabulary or cannot found in GloVe are replaced by the UNK symbol. We set the number of layers of LSTMs to 1 for the hierarchical encoder (for both word encoder and sentence encoder) and the static attention encoder, and 2 for the decoder. The bidirectional LSTMs hidden unit size is set to 500 (250 for each direction). For the LSTM used in the question-based initialier, we use 2 layers unidirectional LSTMs with hidden size 500. The hyperparameters λ_q and λ_a in static attention are initialized as 1.0 and 1.5 respectively. We use dropout with probability $p = 0.3$. All trainable parameters, except word embeddings, are randomly initialized with $\mathcal{U}(-0.1, 0.1)$. For optimization in the training, we use stochastic gradient descent (SGD) as the optimizer with a minibatch size of 32 and the initial learning rate 1.0 for all baselines and our model. We train the model for 100k steps and start halving the learning rate at step 50k, then we halve the learning rate every 10k steps till ending. We set the gradient norm upper bound to 5 during the training. We employ the teacher-forcing training, and in the generating stage, we set the maximum length for output sequence as 15 and block unigram repeated token, the beam size k is set to 50. All hyperparameters and models are selected on the validation set based on the lowest perplexity and the results are reported on the test set.

Baselines and Ablations

We compare our framework with the following baselines and ablations. **Seq2Seq**: the basic encoder-decoder learning framework (Sutskever, Vinyals, and Le 2014) with attention mechanism (Luong, Pham, and Manning 2015). Here we adopt the global attention with general score function. The hidden size of LSTMs for both encoder and decoder is 500. We select the model with the lowest perplexity on the validation set. **HRED**: the **H**ie**R**archical **E**ncoder-**D**ecoder (HRED) with hierarchical attention mechanism. This architecture has been proven effective in several NLP tasks including summarization (Ling and Rush 2017), headline generation (Tan, Wan, and Xiao 2017), and text generation (Li, Luong, and Jurafsky 2015). Here we keep the LSTMs size as 500 for fairness and set the number of the word encoder and sentence encoder layers as 1 and the decoder layer as 2. We employ the question-based initializer for all baselines to generate grammatically coherent distractors. In the generation stage, we follow the same policy and beam size for baselines and ablations during the inference stage to generate three distractors.

		$BLEU_1$	$BLEU_2$	$BLEU_3$	$BLEU_4$	$ROUGE_1$	$ROUGE_2$	$ROUGE_L$
1st Distractor	Seq2Seq	25.28	12.43	7.12	4.51	14.12	3.35	13.58
	HRED	26.10	13.96	8.83	6.21	14.83	4.07	14.30
	Our Model	**27.32**	**14.69**	**9.29**	**6.47**	**15.69**	**4.42**	**15.12**
2nd Distractor	Seq2Seq	25.13	12.02	6.56	3.93	13.72	3.09	13.20
	HRED	25.18	12.21	6.94	4.40	13.94	3.11	13.40
	Our Model	**26.56**	**13.14**	**7.58**	**4.85**	**14.72**	**3.52**	**14.15**
3rd Distractor	Seq2Seq	25.34	11.53	5.94	3.33	13.78	2.82	13.23
	HRED	25.06	11.69	6.26	3.71	13.65	2.84	13.04
	Our Model	**26.92**	**12.88**	**7.12**	**4.32**	**14.97**	**3.41**	**14.36**
Avg. Performance	Seq2Seq	25.25	11.99	6.54	3.92	13.87	3.09	13.34
	HRED	25.45	12.62	7.34	4.77	14.14	3.34	13.58
	Our Model	**26.93**	**13.57**	**8.00**	**5.21**	**15.13**	**3.78**	**14.54**

Table 2: Automatic evaluation results on all systems by BLEU and ROUGE. 1st, 2nd and 3rd distractors are generated under the same policy. The best performing system for each compound row is highlighted in boldface.

Results and Analysis

Automatic Evaluation

Here we evaluate the similarity of generated distractors with the ground truth. We employ BLEU (1-4) (Papineni et al. 2002) and ROUGE (R1, R2, R-L) (Lin 2004) scores to evaluate the similarity. BLEU evaluates average n-gram precision on a set of reference sentences, with a penalty for overly long sentences. $ROUGE_1$ and $ROUGE_2$ is the recall of unigrams and bigrams while $ROUGE_L$ is the recall of longest common subsequences.

Table 2 shows the automatic evaluation results of all systems. Our model with static and dynamic attentions achieve the best performance across all metrics. We can observe a large performance gap between Seq2Seq and models with hierarchical architectures (HRED and our model), which reveals the hierarchical structure is useful for modeling the long sequential input. Another reason could be that some distractors can be generated only use information in several sentences, and sentence-level attentions (both static and dynamic) are useful to emphasize several sentences in the article. Moreover, our model with static attention achieves better performance than its ablation HRED, which shows the static attention can play the role of a soft gate to mask some irrelevant sentences and modulate the dynamic attention.

By comparing the three distractors generated by beam search with a predefined Jaccard distance, we find that the performance drops a little for the second and third distractors. The reason can be two-folds: 1) The second and third distractors have lower likelihood; 2) We set a Jaccard distance threshold as 0.5 to select the second and third distractors, thus they are forced to use some words different from those in the first distractor which is likely to be the best generation.

It is worth to mention that another automatic evaluation method can be applying a state-of-the-art reading comprehension pipeline for RACE to test its performance on our generated distractors. However, the current best performance of such reading comprehension pipeline is only 53.3% (Wang et al. 2018; Zhu et al. 2018; Xu et al. 2017; Tay, Tuan, and Hui 2018), which means half questions in the dataset cannot be answered correctly. Therefore, we do not employ such reading comprehension pipeline to evaluate our generated distractors, instead we hire human annotators to conduct a reliable evaluation, given in the next section.

	Annotator 1	Annotator 2	Annotator 3	# Selected
Seq2Seq	31	35	30	96
HRED	33	40	35	108
Our Model	43	45	36	124
Human	75	70	79	224

Table 3: Human evaluation results. Note that we allow annotators to choose more than one options if the generated outputs are accidentally the same or very semantically similar, therefore, the total number of selected options (552) is larger than the total number of annotated questions (540).

Human Evaluation

We conduct a human evaluation to investigate if the generated distractors can confuse the examinees in the real human test. We employ three annotators with good English background (at least holding a bachelor degree) to answer the MCQs with the generated distractors from different methods. Specifically, for each MCQ, we give 4 distractors as its options: One is a sample from the ground truth, the other three are generated by Seq2Seq, HRED, and our model respectively. Note that we did not give the correct answer option to the annotators, because the current human ceiling performance on RACE dataset is about 95% (Lai et al. 2017). Thus, we need to do a huge amount of annotation for collecting enough questions that are answered wrongly. During the annotation, we told the annotators to select the most suitable option without considering whether there exists a correct option.

For comparison, we count how many times of individual pipelines (the ground truth and three compared methods) are successful in confusing the annotators, i.e. their distractors are selected as answers. We give each annotator 60 articles, and 3 questions per article. In total, we annotated 540 questions, and the results are given in Table 3. We

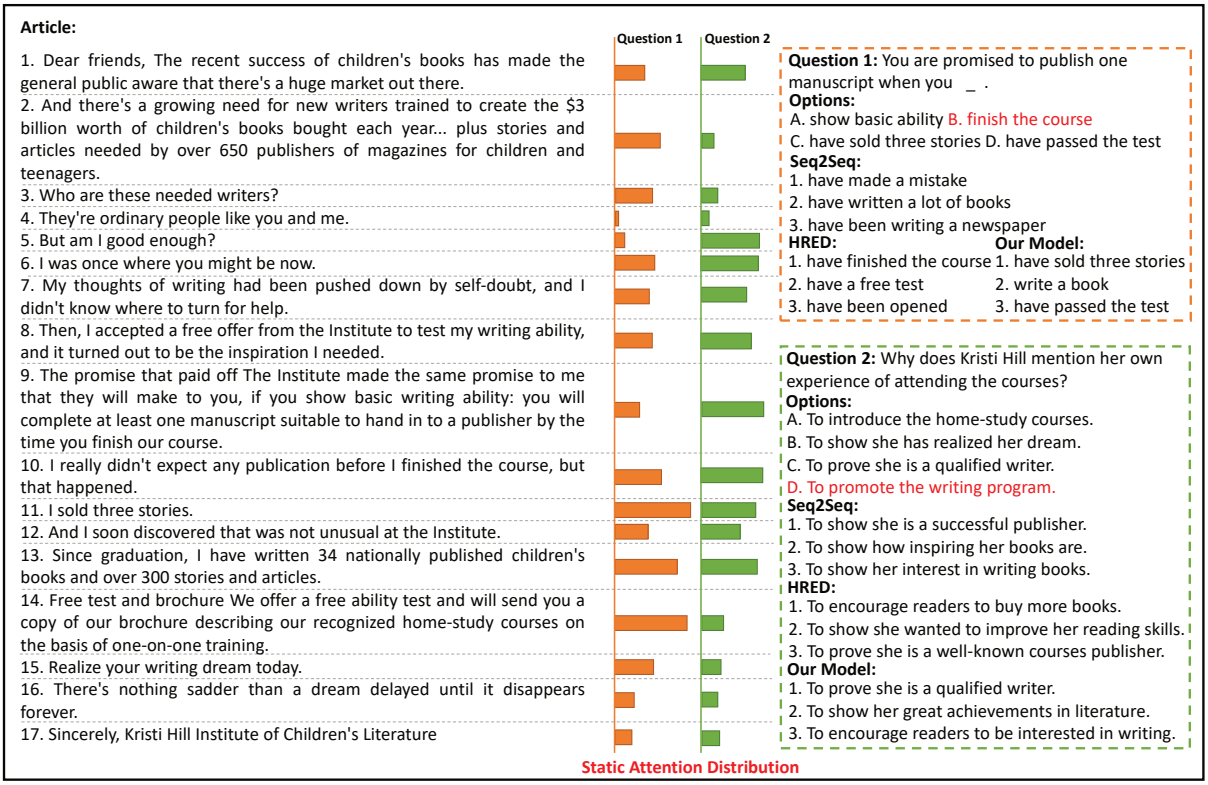

Figure 3: Sample generated distractors. On the right, two example questions are given in dotted lines of yellow and green, and their corresponding static attention distributions are given in the middle by the bars of the corresponding colors.

find that the ground truth distractors (i.e. by "Human") have the strongest capability to confuse the annotators. Among the compared automatic methods, our model performs the best, while Seq2Seq performs the worst, which is a consistent conclusion as drawn from the previous section.

Case Study

In Figure 3, we present some sample distractors generated by human instructors, the Seq2Seq baseline, HRED and our model. To validate the effectiveness of the static attention, we show the static attention distributions over the sentences of the article for the two example questions. The correct options of the questions are marked in red.

Question 1 asks a detailed aspect in the article, which can be directly answered according to the 9th sentence. Since our static attention mechanism suppresses the sentences which contain the answer information, we can see the score for the 9th sentence is relatively smaller than others. The distractor outputs also justify our intuitions. Specifically, the first distractor by **HRED** is semantically identical to the correct option, thus it is not an appropriate distractor. With the help of the static attention, our model does not generate distractors like this. Another effect of the static attention is that it highlights the sentences that are relevant to the question, such as 11th, 13th, and 14th sentences, so that our model can generate better distractors. We can see the distractors generated by our model are semantically relevant to these highlighted sentences. Last but not least, we find that the distractors generated by Seq2Seq baseline either focus on some frequent words in the article such as *publish* and *write*, or contain some completely irrelevant words such as *mistake* and *newspaper*. HRED and our model do not have this problem, because the dynamic hierarchical attention can modulate the word-level attention distribution with the sentence-level attention.

By looking at Question 2, we can also find that the distractors generated by our system are more appropriate and relevant. Because Question 2 requires some inference, it is thus relevant to several sentences across the article. The static attention distribution yields the same conclusion. Specifically, the distribution shows that the 5th to 13th sentences are all relevant to the question, while the 14th sentence which is relevant to the answer option is suppressed. The generated distractors from our system are also semantically relevant to the 5th to 13th sentences.

Conclusions

In this paper, we present a data-driven approach to generate distractors from multiple choice questions in reading comprehension from real examinations. We propose a hierarchical encoder-decoder framework with dynamic and static attention mechanisms to generate the context relevant distractors satisfying several constraints. We also prepare the first dataset for this new setting, and our model achieves the

best performance in both automatic evaluation and human evaluation. For the future work, one interesting direction is to transform this one-to-many mapping problem into one-one mapping problem to better leverage the capability of the sequence-to-sequence framework. Another promising direction could be explicitly adding supervision signals to train the static attention. From the perspective of RACE-like reading comprehension tasks with multiple choice questions, although the performance of existing reading comprehension methods are still quite unsatisfactory, by introducing the distractor generation task, it might open another door for improving the performance, i.e. making adversarial approaches for solving this reading comprehension task possible.

Acknowledgments

The work described in this paper was supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14208815 and No. CUHK 14210717 of the General Research Fund), and Microsoft Research Asia (2018 Microsoft Research Asia Collaborative Research Award).

References

Araki, J.; Rajagopal, D.; Sankaranarayanan, S.; Holm, S.; Yamakawa, Y.; and Mitamura, T. 2016. Generating questions and multiple-choice answers using semantic analysis of texts. In *COLING*.

Chen, W.; Gao, Y.; Zhang, J.; King, I.; and Lyu, M. R. 2018. Title-guided encoding for keyphrase generation. *CoRR* abs/1808.08575.

Chen, C.-Y.; Liou, H.-C.; and Chang, J. S. 2006. Fast - an automatic generation system for grammar tests. In *ACL*.

Goodrich, H. C. 1977. Distractor efficiency in foreign language testing. *TESOL Quarterly* 69–78.

Guo, Q.; Kulkarni, C.; Kittur, A.; Bigham, J. P.; and Brunskill, E. 2016. Questimator: Generating knowledge assessments for arbitrary topics. In *IJCAI*.

Kumar, G.; Banchs, R. E.; and D'Haro, L. F. 2015. Revup: Automatic gap-fill question generation from educational texts. In *BEA@NAACL-HLT*.

Lai, G.; Xie, Q.; Liu, H.; Yang, Y.; and Hovy, E. H. 2017. Race: Large-scale reading comprehension dataset from examinations. In *EMNLP*.

Li, J.; Luong, M.-T.; and Jurafsky, D. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *ACL*.

Liang, C.; Yang, X.; Wham, D.; Pursel, B.; Passonneaur, R.; and Giles, C. L. 2017. Distractor generation with generative adversarial nets for automatically creating fill-in-the-blank questions. In *K-CAP*.

Liang, C.; Yang, X.; Dave, N.; Wham, D.; Pursel, B.; and Giles, C. L. 2018. Distractor generation for multiple choice questions using learning to rank. In *BEA@NAACL-HLT*.

Lin, C.-Y. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Ling, J., and Rush, A. M. 2017. Coarse-to-fine attention models for document summarization. In *NFiS@EMNLP*.

Luong, T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Ma, H.; Lyu, M. R.; and King, I. 2010. Diversifying query suggestion results. In *AAAI*.

Mitkov, R. 2003. Computer-aided generation of multiple-choice tests. *International Conference on Natural Language Processing and Knowledge Engineering, 2003. Proceedings. 2003* 15–.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Sakaguchi, K.; Arase, Y.; and Komachi, M. 2013. Discriminative approach to fill-in-the-blank quiz generation for language learners. In *ACL*.

Stasaski, K., and Hearst, M. A. 2017. Multiple choice question generation utilizing an ontology. In *BEA@EMNLP*.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Tan, J.; Wan, X.; and Xiao, J. 2017. From neural sentence summarization to headline generation: A coarse-to-fine approach. In *IJCAI*.

Tay, Y.; Tuan, L. A.; and Hui, S. C. 2018. Multi-range reasoning for machine comprehension. *CoRR* abs/1803.09074.

Wang, S.; Yu, M.; Chang, S.; and Jiang, J. 2018. A co-matching model for multi-choice reading comprehension. In *ACL*.

Welbl, J.; Liu, N. F.; and Gardner, M. 2017. Crowdsourcing multiple choice science questions. In *NUT@EMNLP*.

Xu, Y.; Liu, J.; Gao, J.; Shen, Y.; and Liu, X. 2017. Dynamic fusion networks for machine reading comprehension. *arXiv preprint arXiv:1711.04964*.

Yang, H.; Xu, Z.; King, I.; and Lyu, M. R. 2010. Online learning for group lasso. In *ICML*.

Yang, H.; Xu, Z.; Ye, J.; King, I.; and Lyu, M. R. 2011. Efficient sparse generalized multiple kernel learning. *IEEE Transactions on Neural Networks* 22:433–446.

Yang, Z.; Hu, J.; Salakhutdinov, R.; and Cohen, W. W. 2017. Semi-supervised qa with generative domain-adaptive nets. In *ACL*.

Yu, A. W.; Dohan, D.; Luong, M.-T.; Zhao, R.; Chen, K.; Norouzi, M.; and Le, Q. V. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *CoRR* abs/1804.09541.

Zhou, T. C.; Lyu, M. R.; and King, I. 2012. A classification-based approach to question routing in community question answering. In *WWW*.

Zhu, H.; Wei, F.; Qin, B.; and Liu, T. 2018. Hierarchical attention flow for multiple-choice reading comprehension. In *AAAI-18 AAAI Conference on Artificial Intelligence*.