Controllable Question Generation via Sequence-to-Sequence Neural Model with Auxiliary Information

Zhen Cao
School of EEE
Nanyang Technological University
Singapore
caoz0008@e.ntu.edu.sg

Sivanagaraja Tatinati
School of EEE
Nanyang Technological University
Singapore
tatinati@ntu.edu.sg

Andy W. H. Khong
School of EEE
Nanyang Technological University
Singapore
andykhong@ntu.edu.sg

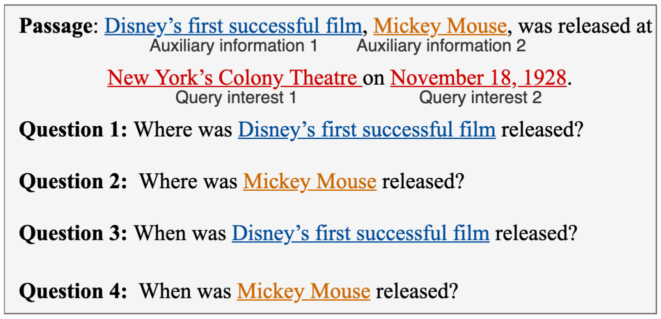
***Abstract*—Automatic question generation (QG) has found applications in the education sector and to enhance human-machine interactions in chatbots. Existing neural QG models can be categorized into answer-unaware and answer-aware models. One of the main challenges faced by existing neural QG models is the degradation in performance due to the issue of one-to-many mapping, where, given a passage, both answer (query interest/question intent) and auxiliary information (context information present in the question) can result in different questions being generated. We propose a controllable question generation model (CQG) that employs an attentive sequence-to-sequence (seq2seq) based generative model with copying mechanism. The proposed CQG also incorporates query interest and auxiliary information as controllers to address the one-to-many mapping problem in QG. Two variants of embedding strategies are designed for CQG to achieve good performance. To verify its performance, an automatic labeling scheme for harvesting auxiliary information is first developed. A QG dataset is also annotated with auxiliary information from a reading comprehension dataset. Performance evaluation shows that the proposed model not only outperforms existing QG models, it also has the potential to generate multiple questions that are relevant given a single passage.**

***Index Terms*—question generation, deep learning, seq2seq, one-to-many, natural language processing**

I. Introduction

Over the past decade, automatic question generation (QG) has gained profound interest in a wide variety of applications including the creation of question-answering datasets [1], [2], generation of practice questions for e-learning platforms [3]–[5], and that of chatbots [6]. Achieving effective QG, however, requires a thorough understanding of a given passage before generating relevant and answerable questions.

An end-to-end sequence-to-sequence (seq2seq) approach that leverages on deep learning techniques has recently been proposed for QG in order to address issues associated with conventional rule-based QG technique [7]. Despite being capable of generating questions with the neural models, most of the generated questions are neither accurate nor answerable. This is due to the fact that, in a human setting, question generation is a one-to-many mapping task, i.e., given a passage (as input) the asker may generate a number of questions with variations in either answers, structure, or nature of the question. Figure 1 shows an example of four possible questions that can be derived from a given passage.


Passage: Disney's first successful film, Mickey Mouse, was released at New York's Colony Theatre on November 18, 1928.

Question 1: Where was Disney's first successful film released?

Question 2: Where was Mickey Mouse released?

Question 3: When was Disney's first successful film released?

Question 4: When was Mickey Mouse released?

Fig. 1. Question formulation is controlled by query interest and auxiliary information.

Due to the above one-to-many mapping problem, deep learning techniques, which are primarily designed for learning one-to-one mapping, yields limited performance. To address ambiguities brought about by the diverse set of possible answers, answer-aware QG models have been developed. Despite these models outperforming answer-unware QG models [8]–[10], the one-to-many mapping problem still exist since questions can also be asked in different ways based on the context that is chosen to form specific questions. For instance, while both questions "Where was Disney's first successful film released?" and "Where was Mickey Mouse released?" are valid (and share the same answer), these two questions are constructed via two different auxiliary information — "Disney's first successful film" and "Mickey Mouse". This illustrative example also highlights the potential of using auxiliary information to increase the specificity of generated questions.

In the context of QG, auxiliary information is defined as the semantic content bearing practical information that flows from the source passage to the target question for the formation of a characteristic utterance [11]. Therefore, asking a question given a passage and an answer is a one-to-many mapping task since different questions can be posed depending on a given auxiliary information. The use of auxiliary information to generate a variety of questions (with the same answer) has several promising applications. For instance, in the design of assessment questions, instructors may wish to prepare different questions for learners within the same topic-of-interest [12]–[14]. In addition, multiple generated questions can enhance diversity when building question-answering datasets, and such diversity can be exploited to train more robust question answering models with better generalization.

As opposed to existing works that do not take auxiliary information into account, we propose a neural model based on auxiliary information and answer information to generate a variety of questions given a passage. This proposed paradigm is named as the controllable question generation (CQG) technique. This technique employs seq2seq with attention [15] and copying mechanism [16] to encode a passage, an answer and any auxiliary information, and to decode the question. We propose two variants of input representation layers that incorporate answer and auxiliary information with different embedding techniques to increase specificity of the generated questions. To further enrich features for encoding, linguistic features including word case, part-of-speech (POS) and named entity tags (NER) are provided to the encoder. The proposed CQG technique is evaluated on the SQuAD dataset annotated with auxiliary information. Results highlight that encoding auxiliary information significantly improves the question generation accuracy.

II. Related Work

Neural network models for QG offer an end-to-end solution that is data-driven and does not rely on hand-crafted rules. Existing neural QG models employ attentive seq2seq model that reads the input text via an encoder before generating the question sequentially via a decoder [1], [7]–[9], [17]–[19]. A technique that learns factual embedding for logic triples before generating question via an attentive gated recurrent unit (GRU) decoder has been proposed [1], while the algorithm presented in [7] generates questions from passages by learning passage-to-question mapping via a seq2seq model with attention mechanism.

It is useful to note, however, that given an input passage, multiple questions can be generated depending on query interests pertaining to the passage. Therefore, QG is a one-to-many mapping task and purely relying on a seq2seq model may not result in good performance [20]. To address this problem, recent works exploit a given query of interest (e.g., answer), and such a model is enriched with an embedding that incorporates an answer position indicator and copying mechanism to copy out-of-vocabulary (OOV) words from the input passage to generate questions [8]. A separate encoder was used to read the answer before fusing with the passage information for question generation [21]. To address challenges associated with generating questions from a long text, a seq2seq model with gated attention encoder and a maxout pointer decoder has been proposed [18]. In addition, target answer in the input passage has been replaced by a token to avoid words from answers being included in the generated questions [10]. A multi-task learning strategy to identify whether a question word should be copied from the input passage or be generated with an overlap between input passage and output question has also been proposed [9].

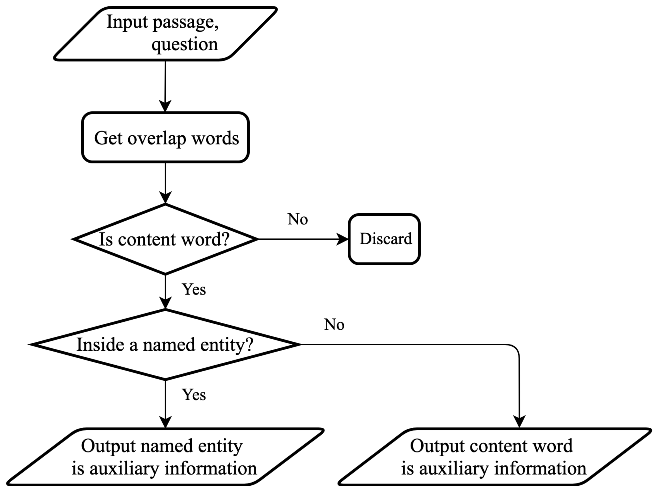


Fig. 2. Automatic labeling scheme for extracting auxiliary information.

III. Methodology

Let P, Q, A, I denote a passage, the question to be generated, answer and auxiliary information, respectively. Both $P = (p_1, p_2, p_3, ..., p_{|P|})$ and $Q = (q_1, q_2, q_3, ..., q_{|Q|})$ are sequences of an arbitrary length, where p_i and q_i denote a word at position i of the sentence or paragraph and question, and $|P|$, $|Q|$ are the number of words in P and Q, respectively. The QG task is defined as that of generating the most probable question $\widehat{Q}$ conditioned on P, A and I, i.e.,

$$\widehat{Q} = \arg\max_{Q} Prob(Q|P, A, I). \tag{1}$$

It is important to note that both the answer and auxiliary information are subset of the given input passage.

A. Dataset Creation

The SQuAD dataset [22] contains more than 100,000 questions and, to the best of our knowledge, there exists no dataset with quadruplets that include passage, question, answer, and auxiliary information. As such, we first define rules for extracting the auxiliary information given the triplets comprising input passage, question, and answer. These rules are derived in accordance to the structure of the question, and auxiliary information in QG should satisfy all the following criteria:

- semantic content carrying practical information;
- occurs in both source passage and target question; and
- must not overlap with words from the provided answer.

To extract auxiliary information for each data point in the SQuAD dataset, the labeling scheme depicted in Fig. 2 is implemented. Given a passage-question pair in the dataset

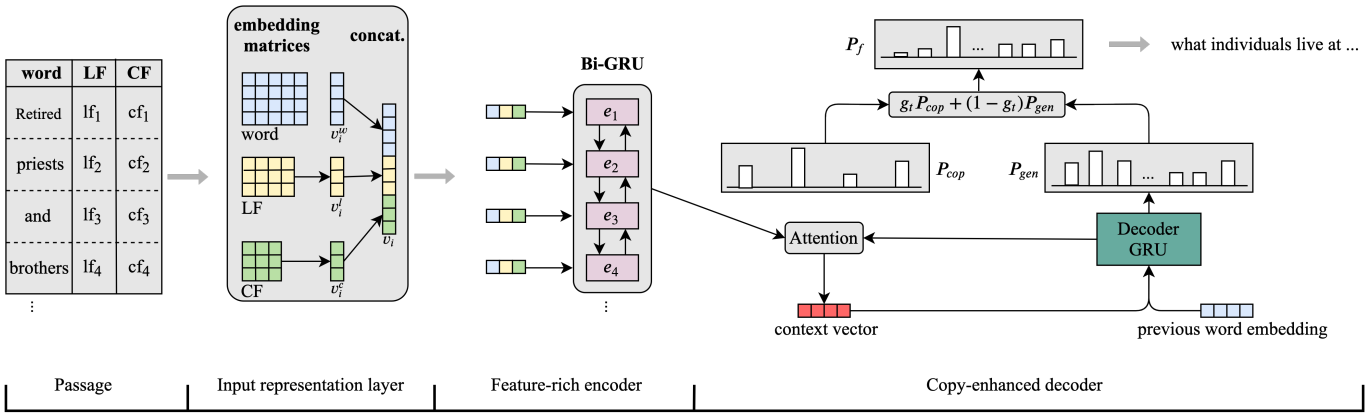


Fig. 3. The proposed controllable question generation (CQG) model, where LF, CF are linguistic and controller features, respectively.

(such as SQuAD), each overlapping word that appears in both passage and question is first identified. From all the identified words, content word—typically a noun, verb, adjective, or adverb—that carries semantic content bearing reference to the world independently of its use within a particular sentence will be retained. Each content word is identified via POS tagging. Heuristically, words with POS "ADJ", "ADV", "NOUN", "NUM", "PROPN" or "VERB" are selected as content word. In the "VERB" tagged content words, all words corresponding to auxiliary verb (e.g., be, do) [23] are rejected since these words can only provide functional or grammatical meaning to the main verb, and they possess minimal or no semantic content. In addition, if the identified content word is labeled as the named entity in the source passage, the named entity is regarded as auxiliary information. The underlying rationale is that context information may be rephrased while posing a question. Therefore, some of the shared semantic content may be missed if one is to limit only to the identical words that only occur in both the passage and question. Otherwise, if the identified content word is not a named entity, the overlapping content words are themselves regarded auxiliary information.

B. Input Representation Layer

With reference to Fig. 3, the architecture of the proposed CQG technique comprises an input representation layer, a feature-rich encoder and a copy-enhanced decoder. The proposed model employs three sets of tokens as the source—input passage, answer and auxiliary information. For a given passage, the answer and auxiliary information control what question should be generated such that the former decides the question type (e.g. what, when, where), while the latter determines the specificity of the expected question. Therefore, both the answer and auxiliary information are regarded as controllers C_A and C_I, respectively.

In this work, we designed a "Y/N" (Yes/No) tagging strategy to the input passage where a word is tagged with a "Y" if it is part of the answer C_A or auxiliary information C_I. We propose two variants of embedding strategies to incorporate C_A and C_I information:

- **Strategy 1:** For passage P, the tag set $\{Y_A, Y_I, N\}$ is applied to mark each word p_i in a passage, where p_i is tagged with Y_A if it is part of C_A, while words belonging to C_I are marked with Y_I, and N represents common words. Therefore, a sequence of Y_A, Y_I, N is obtained to represent what the word at each position is in terms of controllers. This concept is illustrated in the left-most portion of Fig. 3 described by the controller feature (CF) column. A controller embedding matrix is then employed to map each tag as a real-valued vector. For the ith position in a passage P, the tag is mapped to controller embedding w_i^c. This method makes use of the fact that there is no overlap between answer A and auxiliary information I.
- **Strategy 2:** Two separate tag sets $\{Y_A, N\}$ and $\{Y_I, N\}$ are employed to represent the word at each position. Here, Y_A is defined for the word that is being part of C_A and Y_I is the tagger of words belonging to C_I. The C_A and C_I embedding matrices are employed to map two groups of taggers into real-valued vectors independently. Thereafter, two vectors at each position are concatenated to obtain controller embedding v_i^c.

Along with the above controller tagging, each word in the passage is also represented by its linguistic features. For each word in P, feature vectors corresponding to that word and the word vector are concatenated to form a full vector. Each word p_i in the passage is subsequently mapped into a real-valued vector v_i^w via a word embedding matrix $W_{word} \in \mathbb{R}^{d_w \times |V|}$, where d_w is the dimension of word embedding, V is a fixed vocabulary and $|V|$ denotes the number of most-frequent words in the training set. The word embedding matrix is initialized by pre-trained Glove embedding [24], and the out-of-vocabulary words are represented with an OOV token. In addition, word case, POS and NER tags are selected as linguistic features. Similarly, each linguistic feature is mapped to a real-valued vector through an embedding matrix, followed by concatenation to form linguistic features embedding v_i^l for each position in the passage. Finally, word embedding, controller embeddings and linguistic features embedding are concatenated to obtain the final word representation, given as

$$v_i = \left[v_i^w, v_i^c, v_i^l\right]. \quad (2)$$

C. Feature-Rich Encoder

With the input representation layer, input passage $(p_1, p_2, p_3, ..., p_{|P|})$ is converted into a sequence of vectors $(v_1, v_2, v_3, ..., v_{|P|})$. A bidirectional GRU (Bi-GRU) is employed to encode the feature-rich input vectors as a sequence of hidden states $(e_1, e_2, e_3, ..., e_{|P|})$. These states are regarded as the contextualized representations of the input passage. Each hidden state e_t is obtained by concatenating the forward and backward hidden states as

$$\begin{aligned} \overrightarrow{e}_t &= \overrightarrow{GRU}(v_t, \overrightarrow{e}_{t-1}), & (3)\\ \overleftarrow{e}_t &= \overleftarrow{GRU}(v_t, \overleftarrow{e}_{t+1}), & (4)\\ e_t &= [\overrightarrow{e}_t; \overleftarrow{e}_t], & (5) \end{aligned}$$

where v_t, $\overrightarrow{e}_t$ and $\overleftarrow{e}_t$ are the input word representation, the forward and backward hidden states of the tth token in P, respectively.

D. Copy-enhanced decoder

In the decoding stage, a GRU with copying mechanism is employed to generate question words one at a time based on the contextualized representations and the prior decoded question word. At the decoding time step t, the decoder GRU computes the hidden state using

$$d_t = GRU([v_{t-1}^w, c_{t-1}], d_{t-1}), \quad (6)$$

where d_{t-1}, v_{t-1}^w, c_{t-1} are the previous hidden state, previous word embedding and previous context vector, respectively.

The context vector c_t is computed through the additive attention mechanism [15] given by

$$\begin{aligned} s_{t,i} &= k^{\mathsf{T}} tanh(W_d d_t + W_e e_i), & (7)\\ a_{t,i} &= \frac{\exp(s_{t,i})}{\sum_j \exp(s_{t,j})}, & (8)\\ c_t &= \sum_{i=1,...,|P|} a_{t,i} e_i, & (9) \end{aligned}$$

where k^{T}, W_d and W_e are vector and matrices to be learned. Therefore, the probability distribution over the decoder vocabulary is given by

$$\begin{aligned} r_t &= W_r^c c_t + W_r^d d_t + W_r^w v_{t-1}^w, & (10)\\ p(t) &= softmax(W_o r_t), & (11) \end{aligned}$$

where W_r^c, W_r^d, W_r^w and W_o are weight matrices to be learned. Instead of using $p(t)$ for training/generating with the fixed vocabulary, copying words from the source passage is also considered via the copying mechanism [16]. At the decoding time step t, the gate switch probability g_t of copying a word from source sentence (instead of generating a word from the vocabulary) is given by

$$g_t = \sigma(W_g^c c_t + W_g^d d_t), \quad (12)$$

TABLE I
HARVESTING STATISTICS OF AUXILIARY INFORMATION (I)

Item	train set	development set	test set
Original numbers	86,635	8,965	8,964
Missing I numbers	4,207	426	454
Missing I rate	4.86%	4.75%	5.06%

where σ is sigmoid function, W_g^c, W_g^d are the learnable matrices. The "copy" probability distribution P_{cop} over the input word is obtained from (8), where the computed attention weight a_i is regarded as the probability of ith word to be copied. We note that the "generating" probability distribution P_{gen} over the vocabulary is computed using (11). Therefore the final probability P_f distribution over the dynamic vocabulary (i.e., union of original vocabulary and input passage vocabulary) is computed using

$$P_f = g_t P_{cop} + (1 - g_t) P_{gen}. \quad (13)$$

IV. Experiments

A. Dataset and Data Pre-processing

SQuAD is one of the largest reading comprehension datasets derived from Wikipedia with over 100,000 questions. Since the original test set is not publicly available, the development set of SQuAD is randomly split into a development and a test set of equal size. In this work, 82,428 train, 8,539 development and 8,510 test sets comprising sentence-answer-auxiliary information-question samples are harvested. Statistical information pertaining to auxiliary information (that has been extracted via the automatic labelling scheme shown in Fig. 2) is listed in Table I. Auxiliary information is labelled for most of the original samples. This verifies the practicality of the automatic labeling scheme and that there is information dependency between the source passage and the target question in QG.

Before performing experiments on the modified SQuAD dataset, the following pre-processing tasks are performed:

- SpaCy [25] is used to extract linguistic features (POS, NER, word case) for each instance;
- Each word in a sequence is converted to lowercase;
- Tokens SOS and EOS are added to all questions to signify the start and end of question, respectively.

B. Experiment Setup

All the models in this work are implemented in PyTorch 0.4.1 and trained with an Nvidia GTX 1080Ti. For the vocabulary V, the 20,000 most-frequent tokens in the training set are kept. All other tokens outside the vocabulary are replaced by an UNK symbol. Dimension of word embedding is set to 300 and word embeddings are initialized by `glove.840B.300d` pre-trained word vectors. Words that are not contained in GloVe but in the vocabulary are initialized randomly. The controller features are embedded as sixteen-dimensional vectors, and linguistic features including POS, NER, lower case are embedded as eight-dimensional vectors.

TABLE II
EVALUATION RESULTS OF DIFFERENT MODELS BY BLEU 1–4, METEOR AND ROUGE-L

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L
Learn2Ask* [7]	43.09	25.96	17.50	12.28	16.62	39.75
NQG++ [8]	-	-	-	13.27	-	-
s2sa-at-mp-gsa [18]	44.51	29.07	21.06	15.82	19.67	44.24
ASs2s [10]	-	-	-	16.17	-	-
CGC-QG [9]	46.58	30.90	22.82	17.55	21.24	44.53
CQG1	52.90	36.13	26.79	20.56	24.85	49.56
CQG2	54.88	38.50	29.00	22.48	26.09	52.28
CQG1+L	52.96	36.34	27.04	20.75	24.85	49.98
CQG2+L	54.97	38.80	29.42	22.97	26.30	52.60

A single-layer Bi-GRU and GRU of hidden size 300 are used for feature-rich encoder and copy-enhanced decoder, respectively. Dropout with a probability of 0.3 is applied to the models. Hyperparameters are selected by grid search and by comparing with results in the development set.

During training, cross-entropy loss function for question generation is optimized via Adam [26] with a learning rate of 1×10^{-3} and two momentum parameters of 0.8 and 0.999. In addition, optimization is performed across a mini-batch of thirty-two samples and gradient clipping is utilized when gradient norm exceeds 5. During testing, beam search with a beam size of 3 is performed and decoding stops when EOS token that represents the end of sentence is generated. Model parameters are obtained from the best performing model on the development set and its performance is reported on the test set.

C. Evaluation Metrics

All QG models are evaluated with BLEU-n [27], METEOR [28] and ROUGE-L [29]. BLEU measures the overlap of n-gram words between the generated question and the reference at the corpus level, where $1 \leq n \leq 4$. METEOR is based on the precision of uni-gram taking synonyms, stemming and paraphrases into account, while ROUGE-L measures the recall of how word in reference questions appear in generated questions based on longest common sub-sequence (LCS) based statistics. Higher scores denote better generation quality for all metrics.

D. Baselines and CQGs

The baseline neural QG models chosen for comparison with the proposed CQG methods are:

- **Learn2Ask** which is the first neural question generation model that employs attention-based seq2seq model to map a passage to a question [7];
- **NQG++** which is a seq2seq model with an encoder that incorporates answer, POS and NER information [8];
- **s2sa-at-mp-gsa** which contains a gated attention encoder and a maxout pointer decoder to address challenges associated with processing long text inputs [18];
- **ASs2s** which employs two encoders to learn the feature of answer span and the remainder of the passage separately [10];
- **CGC-QG** which is a seq2seq model with an encoder to merge rich features including linguistic information and model generated clue word information [9].

Several variants of the proposed CQG technique are evaluated:

- **CQG1** which is a seq2seq model with both controllers being embedded in the same feature space (Strategy 1);
- **CQG2** which is similar to CQG1 except that controllers are embedded in two different feature spaces (Strategy 2);
- **CQG1+L** which is CQG1 with linguistic features being added;
- **CQG2+L** which is CQG2 with linguistic features being added;
- **CQG-I** where the best model of the aforementioned CQG models is chosen and hereafter named as CQG. In this variant, auxiliary information I is removed from CQG;
- **CQG-A** where answer information A is removed from CQG.

E. Comparison with Baselines

Table II lists the performance of all QG models (evaluated on the SQuAD dataset) in terms of various evaluation metrics. Baselines can be categorized into two types, namely answer-unaware QG models (annotated by * in Table II) and answer-aware QG models. Compared with baselines, CQG models outperform existing models across all metrics; CQG1 surpasses state-of-the-art system CGC-QG by 17.2%, 17.0% and 11.3% in terms of BLEU-4, METEOR and ROUGE-L, respectively. Compared to CQG2+L, the performance improvement increases to 30.9%, 23.8% and 18.1%, for BLEU-4, METEOR and ROUGE-L, respectively. These results affirm that the proposed model which takes both answer and auxiliary information into account can better model QG. This improvement in modeling is due to the fact that providing answers addresses the first-level one-to-many issue (by highlighting the intention of the generated question), while the introduction of the auxiliary information handles the second-level one-to-many issue (by identifying context information that enhances specificity of the generated question).

These results further show that models with linguistic features exhibit modestly higher performance than models that do not consider linguistic information. This is because linguistic

TABLE III
NUMBER OF DIFFERENT TYPES IN EACH LINGUISTIC FEATURE AND NUMBER OF DIFFERENT WORDS

feature	word case	POS	NER	word
number	2	50	18	20,000

feature embedding represents each token from different aspects. In addition, the number of different linguistic features is far smaller than the number of different words (vocabulary size) as shown in Table III. Therefore, it is less challenging to learn patterns from linguistic feature than the word embedding. Although a well-trained word vector may contain information pertaining to linguistic features (such as POS or NER), explicitly concatenating these feature embedding vectors can assist the model by capturing patterns to generate a question more easily.

It is useful to note that the proposed CQG2+L model, which incorporates controller information in two separate feature spaces and takes linguistic features into account, achieves the best performance across all metrics. In addition, CQG2 outperforms CQG1 by 9.3%, 5.0% and 5.5% in terms of BLEU4, METEOR and ROUGE-L, respectively. The fact that CQG1+L performs worse than CQG2+L across all metrics confirms the advantage of Strategy 2. This is expected since it is more challenging for the model to differentiate two controllers information when embedding them within a single feature space. Furthermore, incorporating controller information within two separate feature spaces occupies twice the embedding dimension than in a single feature space resulting in better learning.

F. Ablation tests analysis

Two ablation tests are performed to evaluate the impact of incorporating answer information and auxiliary information in CQG and results are tabulated in Table IV. Without the auxiliary information, the performance of CQG-I is reduced significantly by 9.14, 7.52, and 10.62 in terms of BLEU-4, METEOR and ROUGE-L, respectively. This result is expected and is consistent with that report in [8], [18] since QG models encounter difficulty in modeling semantic content within the passage that otherwise would have been exploited to generate the question.

To highlight the effect of removing query interest, CQG model is trained without answer information in this ablation test. It is interesting to note that while the performance of CQG-A is reduced by 1.89, 1.38, and 2.19 in terms of BLEU-4, METEOR and ROUGE-L, respectively, it still outperforms the state-of-art as presented in [9]. This enhanced performance may be attributed by the fact that additional information arising from auxiliary information is significantly substantial compared to the answer, and that auxiliary information enhances the specificity of the question.

TABLE IV
EVALUATION RESULTS OF ABLATION TESTS

Model	BLEU4	METEOR	ROUGE-L
CQG	22.97	26.30	52.60
CQG-I	13.83	18.78	41.98
CQG-A	21.08	24.92	50.41

V. CONCLUSION

We present results highlighting that QG is, in essence, a two-level one-to-many mapping task. We propose an end-to-end neural framework CQG to model question generation from a passage controlled by query interest and auxiliary information. An automatic auxiliary information labeling scheme is developed to harvest a QG dataset to verify the proposed technique. Experiment results show that our framework improves the performance of question generation and outperforms existing neural QG models.

ACKNOWLEDGMENT

This work was conducted within the Delta-NTU Corporate Lab for Cyber-Physical Systems with funding support from Delta Electronics Inc. and the National Research Foundation(NRF) Singapore under the Corp Lab@University Scheme.

REFERENCES

[1] I. V. Serban, A. García-Durán, C. Gulcehre, S. Ahn, S. Chandar, A. Courville, and Y. Bengio, "Generating factoid questions with recurrent neural networks: The 30M factoid question-answer corpus," in *Proc. Assoc. Comput. Linguistics*, 2016, pp. 588–598.
[2] X. Du and C. Cardie, "Harvesting paragraph-level question-answer pairs from Wikipedia," in *Proc. Assoc. Comput. Linguistics*, 2018, pp. 1907–1917.
[3] M. Heilman and N. A. Smith, "Good question! Statistical ranking for question generation," in *Proc. Human Lang. Technologies: Annu. Conf. North American Chapter Assoc. Comput. Linguistics*, 2010, pp. 609–617.
[4] M. Heilman, "Automatic factual question generation from text," Ph.D. dissertation, Carnegie Mellon University, 2011.
[5] D. Lindberg, F. Popowich, J. Nesbit, and P. Winne, "Generating natural language questions to support learning on-line," in *Proc. Eur. Workshop Natural Lang. Gener.*, 2013, pp. 105–114.
[6] N. Mostafazadeh, I. Misra, J. Devlin, M. Mitchell, X. He, and L. Vanderwende, "Generating natural questions about an image," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2016, pp. 1802–1813.
[7] X. Du, J. Shao, and C. Cardie, "Learning to ask: Neural question generation for reading comprehension," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2017, pp. 1342–1352.
[8] Q. Zhou, N. Yang, F. Wei, C. Tan, H. Bao, and M. Zhou, "Neural question generation from text: A preliminary study," in *Proc. Nat. CCF Conf. Natural Lang. Proc. and Chin. Comput.*, 2017, pp. 662–671.
[9] B. Liu, M. Zhao, D. Niu, K. Lai, Y. He, H. Wei, and Y. Xu, "Learning to generate questions by learning what not to generate," in *Proc. World Wide Web Conf.*, 2019, pp. 1106–1118.
[10] Y. Kim, H. Lee, J. Shin, and K. Jung, "Improving neural question generation using answer separation," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 6602–6609.
[11] E. Lepore and B. C. Smith, *The Oxford Handbook of Philosophy of Language*. Oxford University Press, 2009.
[12] N. R. Council *et al.*, *Learning and understanding: Improving advanced study of mathematics and science in US high schools*. National Academies Press, 2002.
[13] S. Supraja, K. Hartman, S. Tatinati, and A. W. H. Khong, "Toward the automatic labeling of course questions for ensuring their alignment with learning outcomes," in *Proc. Educational Data Mining*, 2017, pp. 56–63.

[14] S. Supraja, S. Tatinati, K. Hartman, and A. W. H. Khong, "Automatically linking digital signal processing assessment questions to key engineering learning outcomes," in *Proc. IEEE Intl. Conf. Acoust., Speech, Signal Process.*, 2018, pp. 6996–7000.

[15] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 1412–1421.

[16] C. Gulcehre, S. Ahn, R. Nallapati, B. Zhou, and Y. Bengio, "Pointing the unknown words," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2016, pp. 140–149.

[17] L. Song, Z. Wang, W. Hamza, Y. Zhang, and D. Gildea, "Leveraging context information for natural question generation," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Lang. Technol.*, 2018, pp. 569–574.

[18] Y. Zhao, X. Ni, Y. Ding, and Q. Ke, "Paragraph-level neural question generation with maxout pointer and gated self-attention networks," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 3901–3910.

[19] X. Sun, J. Liu, Y. Lyu, W. He, Y. Ma, and S. Wang, "Answer-focused and position-aware neural question generation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 3930–3939.

[20] Y. Gao, L. Bing, W. Chen, M. Lyu, and I. King, "Difficulty controllable generation of reading comprehension questions," in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 4968–4974.

[21] W. Hu, B. Liu, J. Ma, D. Zhao, and R. Yan, "Aspect-based question generation," in *Proc. Int. Conf. Learning Representations, (ICLR) Workshop Track*, 2018.

[22] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2016, pp. 2383–2392.

[23] P. W. Culicover, *Natural language syntax*. OUP Oxford, 2009.

[24] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2014, pp. 1532–1543.

[25] M. Honnibal and I. Montani, "spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing," *To appear*, 2017.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learning Representations, ICLR*, 2015.

[27] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2002, pp. 311–318.

[28] M. Denkowski and A. Lavie, "Meteor universal: Language specific translation evaluation for any target language," in *Proc. Workshop on Statistical Mach. Transl.*, 2014, pp. 376–380.

[29] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Proc. Text Summarization Branches Out*, 2004, pp. 74–81.