Arabian Journal for Science and Engineering (2024) 49:3027–3041
https://doi.org/10.1007/s13369-023-07840-7

RESEARCH ARTICLE-COMPUTER ENGINEERING AND COMPUTER SCIENCE

Automated Question and Answer Generation from Texts using Text-to-Text Transformers

Rupali Goyal[1] · **Parteek Kumar**[1] · **V. P. Singh**[1]

Received: 8 September 2022 / Accepted: 20 March 2023 / Published online: 3 May 2023

Abstract
Automatic question generation and automatic question answering from text is a fundamental academic tool that serves a wide range of purposes, including self-study, coursework, educational assessment, and many more. Manual construction of questions is a time-consuming and complicated process that requires experience, whereas automating the process diminishes the costs of manual question creation and fulfills the need for a persistent supply of questions for the tutors and self-evaluators. This paper uses an encoder–decoder architecture-based text-to-text transfer transformer (T5) intending to generate several types of question–answer pairs over a given context, including subjective question–answers having short and long answers, fill-in-the-blanks-type question–answers, Boolean answer (yes-or-no)-type questions, and multiple-choice question-answers. The model has been evaluated on benchmark datasets—SQuAD, QuAC, and BoolQ; over automated metrics—BLEU, ROUGE, METEOR, F1, and accuracy. The model outperformed previous baseline models, with 18.87 and 25.24 scores over BLEU-4 and METEOR metrics, respectively. The paper also demonstrates that the proposed system efficiently generates question–answer pairs where the baseline approaches struggled. The evaluation analysis also shows that the generated question–answer pairs are comparable with existing systems and even better in terms of diversity. Also, the generated questions are grammatically and contextually correct, and the answer generated matches the question in the textual context.

Keywords Question answering · Automatic question–answer generation · Multiple-choice question–answers · Fill-in-the-blanks · Subjective question–answer · Boolean answer–questions

1 Introduction

Automatic question generation and question answering is critical and plays a key role in many application domains, such as education [1–3], personal assistance [4], and healthcare [5]. Manual question generation requires a lot of effort and resources and is costly and time-consuming. Traditional classrooms involve periodic tests, quizzes, and exams, along with the impromptu questions asked by the instructor during or after every session. This enables the learner to gauge their understanding and the instructor to gauge the effectiveness of their lessons. But creation and selection of questions is a time-consuming task. Creating good-quality questions is a complex process that requires training and experience. Also, due to the unprecedented convenience of Massive Open Online Courses (MOOCs), a large number of students have shifted to this self-learning paradigm of education to learn new concepts or aid their classroom courses. However, most online classrooms lack relevant and sufficient exercises to test the students due to the paucity of time on the creator's side. Therefore, this becomes the need of the hour. As a result, there has been much interest in building an autonomous system for generating questions and corresponding answers during the last decade.

Questions are an integral part of learning and a fundamental tool in education. They can be used for querying more information or making sure that the class is engaged. Questions provide learners with feedback about their understanding and misconceptions, offer the opportunity to practice retrieving information from memory, highlight the important learning material and help learners focus on it, motivate learners to engage in learning activities, and repetition of core concepts for reinforcement learning are few of the advantages of asking questions. Questions are broadly divided into

✉ Rupali Goyal
rrupali20_phd17@thapar.edu

[1] Computer Science and Engineering Department, Thapar Institute of Engineering and Technology, Patiala, Punjab, India

two categories: objective questions and subjective questions [6]. The objective question asks individuals to choose the correct response from two to four alternatives or to provide a word/phrase to complete a sentence or to answer a question. The most prevalent types of objective question in education are multiple-choice, true–false, and fill-in-the-blank. The subjective questions, on the other hand, demand response in terms of explanation that enables individuals to construct and create a response in their wording. Long and short answer-type questions are the two well-known examples of subjective questions. However, for an effective assessment, both subjective and objective questions are necessary, and the proposed system has the capability of generating these questions.

The answers are extracted based on the information accumulated from the given context. The answers are categorized based on different tasks. For objective questions, the output is a word or phrase from context, while the multiple-choice approach requires selecting the correct answer from a list of potential answers. In the subjective question answering technique, a subsequence of the provided context is extracted as a response for questions with the short answer, and for long answer-type questions, the free answering techniques are used.

Traditionally, rule-based approaches and statistical approaches [13, 14]. The creation of rules and templates is extremely expensive, lacks diversity, and is hard to generalize on different domains. The recent advancements in deep neural networks have come to be known to outperform in most areas of natural language generation in the past, but with an overhead cost of processing and a considerable amount of training. However, with the decrease in hardware cost and increased availability (backed by numerous cloud platforms), the transition to neural techniques has been quite favorable. The developments in deep neural networks, such as memory networks [7], attention mechanisms [8], and copy mechanisms [9], have shown promising results for the question and answer generation task. However, generating diverse question–answer pairs from the text remains a significant challenge. This paper addresses this challenge and has proposed a system to automatically generate a variety of question–answer pairs over a given passage.

These approaches required sequential processing, making training a very tedious and time-consuming task, but transformers [10] allowed parallelization of tasks by taking the entire sequence as input instead of token by token which made them popular. Transformer-based [10] networks downstream on a specific task [11] followed by fine-tuning over a large corpus have become the norm for a variety of natural language tasks. Several variations of transformers have been introduced in the past years. Transformers like GPT [12] are based on a left-to-right decoder whereas BERT [11] is based on a deep bidirectional long short-term memory encoder. On the other hand, BART [13] and T5 [14] transformers use encoder–decoder architecture. This paper adopts the conventional transformer-based sequence-to-sequence structure. The proposed framework is built upon a text-to-text transfer learning model [14]. The training objective of this model is to generate an end-to-end question–answer pair generation based on the given context. The key objective of this paper is the use of T5 to automatically generate sentence and paragraph-level question–answer pairs with diverse perspectives over a given context. Figure 1 shows the overview of the proposed question–answer pair generation system.

The proposed question–answer pair generation system takes a context passage as input and creates a list of questions on content knowledge with the extracted answers as output. The approach starts with loading textual context over which question–answer pairs have to be generated. The text uploaded can be a single-sentence or multi-sentence passage. This text is then passed to the next block, where it is pre-processed and split into sentences using a sentential tokenizer. These text tokens are then fed to the next block to obtain the position of each sentence in the context. These positions are attained using positional embeddings. Then, the sentence which contains answers is realized and highlighted using token and task prefixes are appended to the context with the highlighted sentence. The length padding or truncation is performed before applying the fine-tuned T5 model for answer extraction. After applying the model, a list of answers has been obtained and joined. The embeddings for these extracted answers are obtained along with their corresponding positional embeddings. The task prefixes are appended and the length padding or truncation is performed. Then, the fine-tuned T5 model has been applied to generate the questions that matched the text in the context. Finally, a list of question–answer pairs has been obtained that are grammatically and contextually correct. These generated question–answer pairs facilitate teachers in developing instructional content for various domains. Also, automating the process of tests creation helps tutors for academic purposes and students with their self-evaluation.

The rest of this paper is organized as follows: Section 2 presents a brief overview of the existing approaches to question generation and answer extraction tasks. The architecture of the question–answer generation system and the design of each module is discussed in Sect. 3. The algorithms behind the implementation of each module are also explored. The evaluation results obtained for the system are presented in Sect. 4. Section 5 draws the conclusion along with the future scope of work.

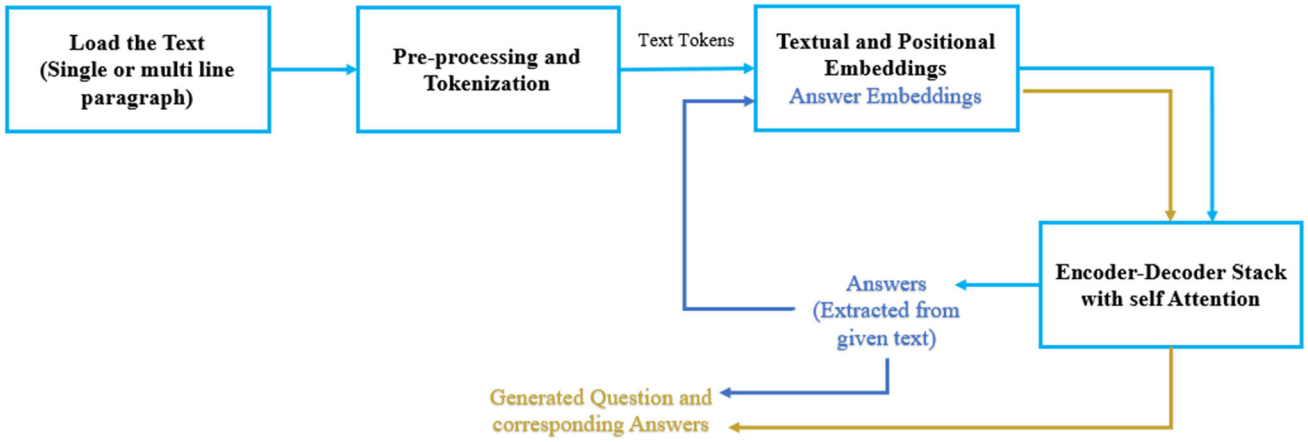


Fig. 1 Block diagram for automatic question–answer pair generation from text

2 Related Work

Traditionally, a rule-based approach was suggested to construct question templates and manually create questions from a given text [15, 16]. The creation of rules and templates is extremely expensive, lacks diversity, and is hard to generalize on different domains. Recently, deep neural network-based approaches [17, 18] have been proposed that address the issues of rule-based approaches to generate question-answers without handcrafting rules. These approaches include recurrent neural networks (RNNs) [19] which encompass long short-term memory (LSTM) networks [20], gated recurrent units (GRUs) [21] and several variations of these. Therefore, there has been a clear shift from rule based and statistical methods of natural language processing to deep learning methods and now to transformers due to the excellent results obtained from the latter. Neural networks have been known to outperform in most areas of natural language generation in the past, but with an overhead cost of processing and a considerable amount of training. However, with the decrease in hardware cost and availability (backed by numerous cloud platforms), the transition to neural techniques has been quite favorable. With advancements like memory networks [7], attention mechanisms [8], and copy mechanisms [9], deep neural networks have also shown promising results for the question–answer generation task.

Du et al. [19] pioneered the automated generation of questions using a deep sequence-to-sequence neural model. For answer-aware question generation, firstly, the positions of the answer span are extracted from the input sentence, and then, the answer-specific questions are generated. Most existing studies [19, 22] use an encoder–decoder framework with an attention mechanism [23, 24]. However, different strategies have been incorporated for answer information by different models, such as first detect the question-worthy answer and then generating the answer-aware question [25]. Similarly, other strategies like embedding the relative distance between the answer and the context words [26], key-phrase extractor for the key answer candidates [27–29], answer position indicator [8, 30], and so on are also incorporated. Du and Cardie [23] used gated coreference knowledge for paragraph-level answer-aware question generation. Zhao et al. [31] proposed a seq2seq network having a gated self-attention encoder and a maxout pointer decoder for paragraph-level single question generation. Nema et al. [32] had proposed refine networks for the question generation task. Kim et al. [33] have used deep neural networks to generate questions using answer-separation. Lopez et al. [34] have used a decoder-only transformer for paragraph-level question generation. Liu [35] have used seq2seq-based algorithms with copy mechanism and attention mechanism for question generation. There have utilized different approaches for question–answer generation task but are limited in capabilities as compared to our proposed approach.

The introduction of attention gave way to transformers [21]. All previous approaches required sequential processing, making training a very tedious and time-consuming task, but transformers allowed parallelization of tasks by taking the entire sequence as input instead of token by token, making them popular. Several variations of transformers have been introduced in the past years. Transformers like GPT [12] are based on a left-to-right decoder, whereas BERT [11] is based on a deep bidirectional long short-term memory encoder. On the other hand, BART [13] and T5 [14, 36] transformers use encoder–decoder architecture. The current state of the art leverages transformer-based models [10, 11, 13, 37] for question–answer generation. Transfer learning models [38] focus on preserving knowledge obtained from training on one problem and applying it to a related but different problem [11, 14]. Dehghani et al. [39] use transformer architecture for open-domain question answering. Lopez et al. [34] used

GPT-2 transformer architecture to generate questions over a given paragraph.

Most of the existing question–answer generation models had been proposed over a single-sentence context and can generate only three types of WH-question (what, who, where). These systems use handcraft rules for question–answer generation and falter on complex sentences [40]. Also, the existing neural models generate one question per context [32]. The previous works on question generation also use transformers [34], but it is either encoder or decoder based. Some research works use encoder–decoder-based approach but lack diversity of the generated question–answer pairs [36].

In this paper, an encoder–decoder architecture-based transformer model [14] has been used for generating question–answer pairs given context. The context can be a single-sentence or multi-sentence passage. The proposed system has the capability to scale well over different perspectives and styles of question–answers. The creation of a system to facilitate teachers and students in generating instructional content for various domains, which can serve multiple uses but also consume a lot of time when produced manually, is the primary motivation of this work. For an effective assessment, both subjective and objective questions are necessary, and the proposed system has the capability of generating these questions. A significant problem faced by instructors engaged in frontline instruction of classes is the lack of time for creating good-quality instruction content. The generation of diverse and grammatically correct question–answer pairs from the text remains a significant challenge. This paper addresses these challenges and has proposed a system to automatically generate a variety of question–answer pairs over a given passage.

2.1 Contributions of the Paper are as Follows

The proposed question–answer pair generation system automatically generates sentence and paragraph-level question–answer pairs with diverse perspectives over a given context. The main contributions of this paper are as follows:

- A system that acts as a one-stop destination for generating subjective as well as objective-type questions has been proposed.
- The proposed system is capable to handle fill-in-the-blank, multiple-choice, Boolean, and long/short answers.
- This work fine-tunes a text-to-text transfer learning (T5) transformer to generate question–answer pairs over three large-scale benchmark datasets: SQuAD, QuAC, and BoolQ.
- The proposed system has been evaluated over automated metrics, such as BLEU, ROUGE, METEOR, F1 score, and accuracy, and has outperformed state-of-the-art baselines over BLEU-4 and METEOR metrics with a score of 18.87 and 25.24, respectively.
- A comparative analysis has been performed with the existing models to show the effectiveness of our proposed question–answer generation system.
- The questions generated by our approach are grammatically and contextually correct and the answer generated matches the question in the textual context.

3 Proposed Model Architecture

The proposed approach utilizes a stacked encoder–decoder with text-to-text transfer learning to generate various question–answers. Figure 2 illustrates that the T5 model has been fine-tuned using different settings on multiple tasks, such as QA, answer extraction, and QG.

The task has been formally described using Eq. (1). Let t denote the text, a denote the answer which is in span of c and let q denote the question targeting the answer a. Given a text t, the task is to model

$$P(q,\ a|t)\ =\ P(a|t)\ P(q|a,\ t) \tag{1}$$

The first part of the equation model $P(a|t)$ as an answer extraction task. In this way, the question generation task has been carried out without explicitly providing answers and the model has been trained to search for answers in the given text. For this task, text and answer pairs have been used where text is used as an input and answers are used as targets for training an answer extraction task. For the second part of the equation, $P(q|a, t)$, the answer and text pairs are utilized as inputs and targeted question for given answer as the target in training. In the generation part, the answer extraction is done before question generation.

The framework for the automatic question generation and question answering is shown in Fig. 3. The proposed approach utilizes transfer learning with a text-to-text transfer transformer (T5) that can handle long-term dependencies well. This T5 model [14] is a unified framework-based encoder–decoder model that converts every problem into a text-to-text format. It has been trained on a variety of supervised and unsupervised tasks. For unsupervised tasks, it has been trained on Colossal Clean Crawled Corpus which is a novel 750 Gigabyte-huge dataset, and for supervised tasks, several well-known datasets were utilized.

Deep transformer-based [10] networks downstream on a specific task [11] followed by fine-tuning over a large corpus have become the norm for a variety of natural language tasks. In this paper, the task of question–answer generation is addressed. Specifically, when given a context, the model is entrusted with generating appropriate questions and corresponding answers to the generated question.

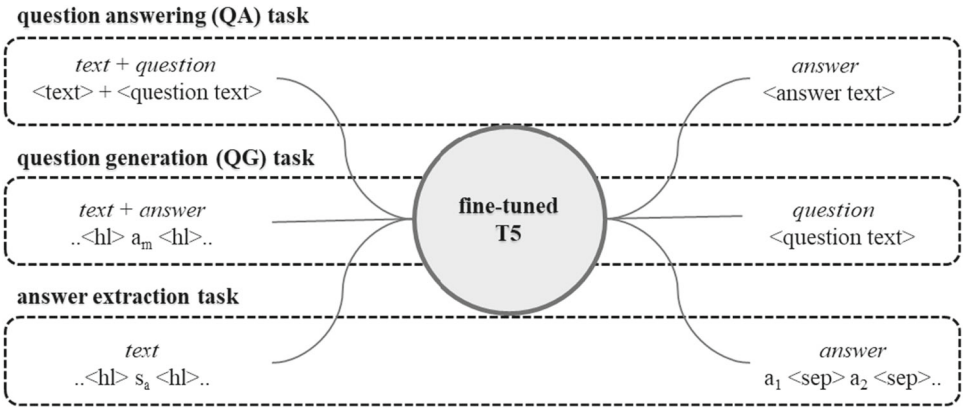


Fig. 2 Multi-task fine-tuning of the pre-trained T5 model: (1) QA task uses text and question pair as input and generates answer as output, (2) QG task uses answer highlighted text as input and generates question as output, and (3) answer extraction task uses sentence highlighted text as input and extract list of answers with separator as output

The context input is first aligned as a specific input sequence by adding a special token [CLS]. The context input sequence can either be a single sentence or a group of sentences. A special token [SEP] is introduced between the tokens of two consecutive sentences to separate information from different sentences. In addition, a learned embedding is added to every token to denote whether it belongs to which sentence. The sum of token embeddings and position embeddings is the input representation of a given token. The first encoder in the encoder-stack receives these embeddings of the input sequence. The encoder then propagates and transforms the resultant to the next encoder. To get a better understanding of a certain word in the sequence, a self-attention layer is used. This layer allows the model to look at the other words in the input sequence. All the decoders in the decoder-stack receive the output from the last encoder in the encoder-stack. The encoder–decoder attention layer in the decoder-stack enables the decoder in focusing on the pertinent segments of the input sequence. The keywords or keyphrases are obtained as output. These keywords are then mapped with context sentences to obtain corresponding questions. The proposed system finally outputs the extracted answers and generated questions as question–answer pairs. Figure 4 represents the flow diagram of this approach.

The flow of the approach starts with loading textual context over which question–answer pairs have to be generated. The context is then split into sentences using a sentential tokenizer. The position of each sentence in the context has been attained using positional embeddings. Then, the sentence which contains answers has been realized and highlighted using < *hl* > token. After this, a task prefix (extract answer) has been appended to the context with the highlighted sentence. Prior to applying the fine-tuned T5 model for answer extraction, length padding or truncation was done. After applying the model, a list of answers has been obtained, which are joined using < *sep* > token. These answer embeddings and positional embeddings are obtained. After this, a task prefix (generate questions) has been appended to the context and the length padding or truncation is done. Then, the fine-tuned T5 model has been applied to generate the questions that match the text in the context. Finally, a list of question–answer pairs has been obtained that are grammatically and contextually correct. The architectural diagram of each encoder and decoder layer used is shown in Fig. 5.

The encoder and decoder component consists of a stack of six encoder layers on top of each other and a stack of the same number of decoder layers. Each encoder comprises two sub-layers: a self-attention layer and a feed-forward layer. The input to the encoder is first fed to a self-attention layer. This layer helps the encoder look at all the words in the input sentence as it encodes a specific word. The normalization layer is applied to each subcomponent layer where the activations are rescaled. A residual skip connection adds each sub-layer's input to its output. Within the feed-forward layer, a dropout is applied on the attention weights, skip connection, and at the input and output of the stack. The resultant outputs are fed to a feed-forward neural network (the second sub-layer of the encoder). The same feed-forward network is independently applied to each position and repeats the dropouts and normalization. The decoder has similar layers to the encoder and an additional attention layer between them. This additional attention layer helps the decoder focus on relevant parts of the input sentence.

3.1 Implementation Details

This paper uses the pre-trained T5-small model with 60 M parameters. The most notable feature of the used transfer learning model is its *text-to-text* nature. This text-to-text nature of the model enables it to learn any natural language task without altering the loss functions and hyperparameters. All these models are trained using the same hyperparameters with different data preparation. All the training and data preparation has been done on Google Colab. Pytorch is used for training and developing neural models. The grid-search technique had been utilized for identified hyperparameters, including learning rate, optimizer type, and the number of training epochs. And we have selected the set of parameters

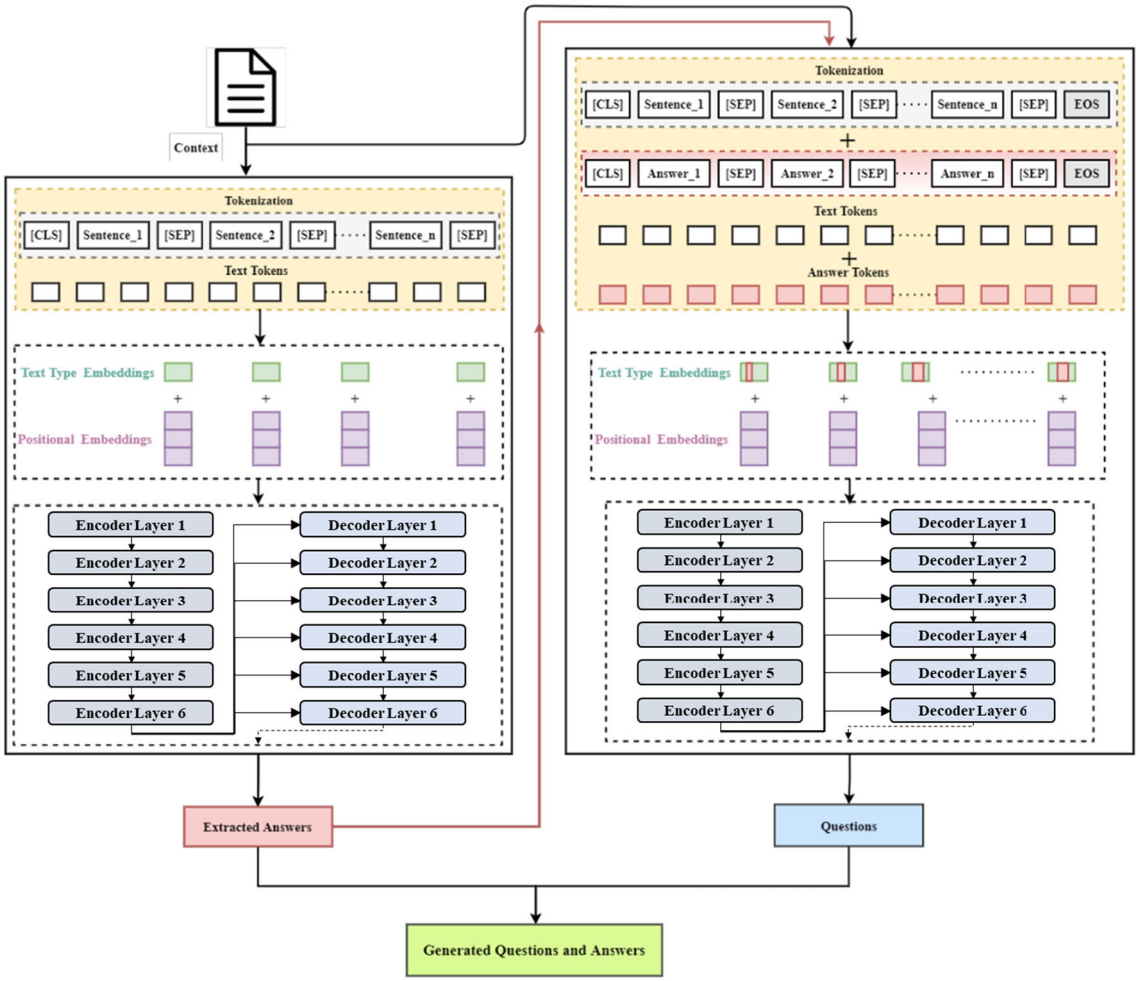
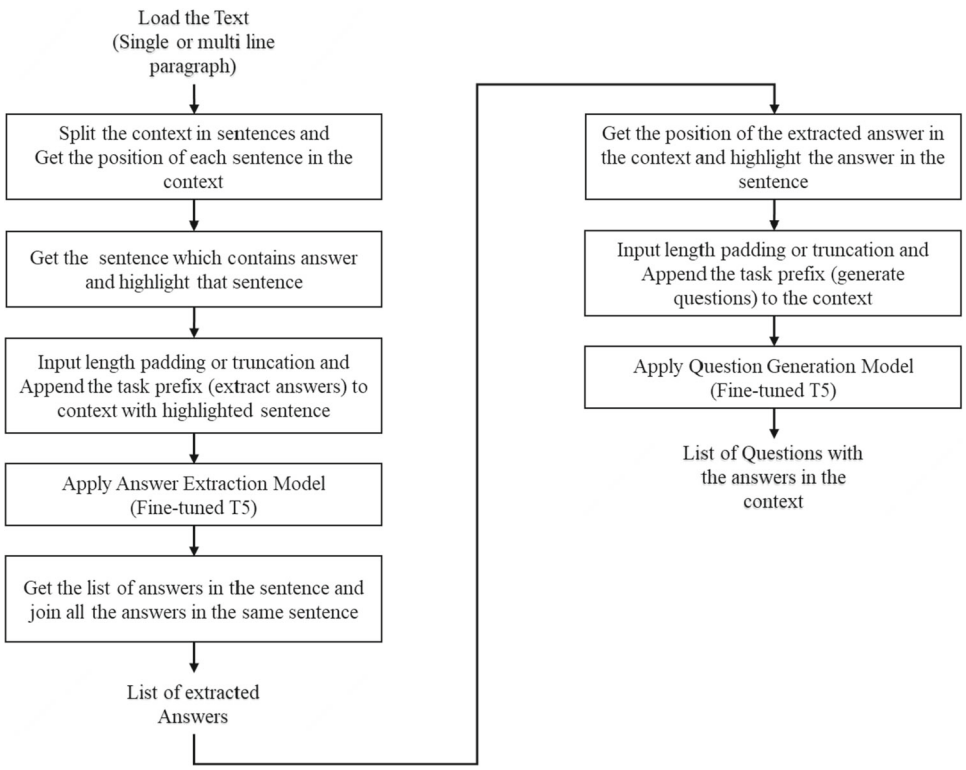


Fig. 3 Framework for automatic question and answer generation from text

that had attained the overall best scores in all metrics. The batch size for training and evaluation is 32. The model is trained with a learning rate of 1e−4 for 18 epochs. The beam search technique is used for sequence decoding with a beam of size 4. The experiments described are implemented using Hugging Face's Transformer library [41]. For fine-tuning a special class, the *trainer* class is used which simplifies and abstracts the complex training procedure and is optimized for training transformer models. The training arguments are decided according to the model to be trained, and a trainer is instantiated with those arguments. All models trained for the system use the same training script but different datasets and different training parameters. The algorithms used in the prediction phase for different models in the form of pseudo-codes along with explanations are mentioned below.

3.1.1 Subjective Question–Answer (Long and Short) Generation

For subjective question–answer generation, the process is divided into two tasks. The first task is to seek potential answers from the text given. The *Answer-Extractor* model is used for this task, which has been trained to extract answers from a given context. Each sentence from the input text is recognized as a separate input. These sentences are then formatted to be preceded by the label *extract answers* and passed to the *generate* function, which feeds these formatted texts to the model and retrieves the answers. For the second task, the extracted answers from the first task are mapped to the original context sentences, and a highlight token $< hl >$ is added at the start and the end answer positions. This mapped text with highlight tokens is then formatted to contain the *generate question* label which is then passed to the *generate* function of the model for generating questions according to the highlighted tokens. Finally, a list of the generated question with corresponding answers is obtained. The pseudo-code for subjective question–answer generation is given below.



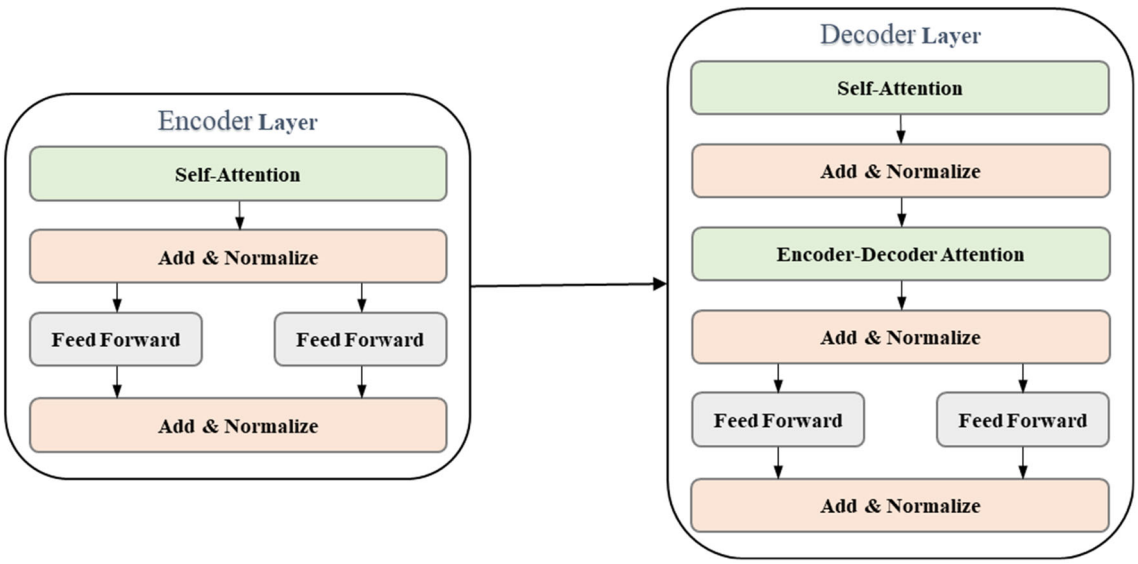


Fig. 4 Flow diagram for automatic question and answer generation approach

Fig. 5 Architecture of each encoder and decoder Layer

Algorithm 1: Subjective Question-Answer Generation (Long and Short answers)

Input: Contextual Text, context; **Output:** Question-Answer pairs

```
Begin
    context = get_input_text()
    sentences = sent_tokenize(text)
    for sentence in sentences do
        answer_input = "extract answers:" + {sentence} + "</s>"
        generated_answers = add_to_list(GenerateAnswers(answer_input))
    end
    for answer in generated_answers do
        find answer's start and end positions in input context
        qg_input = "generate_question:" + context[ :start_position - 1] +
        "<hl>" + context[ start_position : end_position ] + "<hl>" +
        context[ end_position + 1 :] + "</s>"
        generated_questions = add_to_list( GenerateQuestions(qg_input)
        append generated_questions and answer to output_dict
    end
    reformat output_dict
End
```

3.1.2 Fill-in-the-Blanks-Type Question–Answer Generation

For the generation of fill-in-the-blanks-type question–answers, the first task is to recognize the potential spots for introducing blanks in the sentences. This is done using the *Answer-Extractor* model, which has been trained to extract answers from a given context. Each sentence from the input text is formatted by appending the label *extract answers* and then passed to the *generate* function, which feeds these formatted texts to the model and retrieves the answers. The answers are then mapped back to the original input context, and dashes or blanks are introduced at the positions of the answer phrase. Finally, both the generated fill-in-the-blank and its answer phrase are appended together as output. The below mentioned is the pseudo-code for fill-in-the-blanks generation.

Algorithm 2: Fill-in-the-Blanks type Question Answer Generation

Input: Contextual Text, context; **Output:** Sentences with blanks and answer phrases for the blanks

```
Begin
    context = get_input_text()
    sentences = sent_tokenize(text)
    for sentence in sentences do
        answer_input = "extract answers:" + {sentence} + "</s>"
        generated_answers = add_to_list( GenerateAnswers(answer_input))
    end
    for answer in generated_answers do
        find answer's start and end positions in input context space = '_'
        * len(answer)
        append generated_fibs and answer to output_dict
    end
    reformat output_dict
End
```

3.1.3 Boolean Answer–Question Generation

For generating Boolean questions having true/false or yes/no-type answers, firstly the input text is sentence tokenized. Each sentence from the input text is recognized as a separate input. These sentences are then formatted such that each sentence is appended with the *Boolean* label. These labeled data are then passed to the *generate* function, which feeds these formatted texts to the model to generate Boolean-type questions. The pseudo-code for Boolean answer–question generation is attributed below.

ated along with the correct answer. For generating distractors, the *Sense2Vec* module is used. The extracted answers having considerable equivalent distractors are selected from the list of extracted answers and any answer which do not have a specific number of distractors is discarded. The selected answers are mapped to the original context sentences, and a highlight token < *hl* > is added at the start and the end answer positions. This mapped text with highlight tokens is then formatted to contain the *generate question* label which is then passed to the *generate* function of the model for generating questions according to the highlighted tokens.

Algorithm 3: Boolean Answer Question Generation

Input: Contextual Text, context; **Output:** Boolean Questions having yes/no or true/false answer

```
    Begin
1:      context = get_input_text()
2:      sentences = sent_tokenize(text)
3:      for sentence in sentences do
4:          boolq_input = "boolean:" + {sentence} + "</s>"
5:          generated_bool_questions=
            add_to_list(GenerateBoolQuestions(boolq_input))
6:          append generated_bool_questions to output_dict
7:      end
8:      reformat output_dict
    End
```

3.1.4 Multiple-Choice Question–Answer Generation

For generating multiple-choice question-answers, the first task is to seek potential answers from the text given. Each sentence from the input text is recognized as a separate input. These sentences are then formatted to be preceded by the label *extract answers* and passed to the *generate* function, which feeds these formatted texts to the model and retrieves the answers. The distractors are also to be gener-

The final output contains the generated question, correct answer, and a list of distractors. The pseudo-code for multiple-choice question–answer generation is given below.

Algorithm 4: Multiple Choice Question Answer Generation

Input: Contextual Text, context; **Output:** Questions with one correct answer and a list of distractors

```
Begin
    context = get_input_text()
    sentences = sent_tokenize(text)
    for sentence in sentences do
        answer_input = "extract answers:" + {sentence} + "</s>"
        generated_answers = add_to_list( GenerateAnswers(answer_input))
    end
    for answer in generated_answers do
        distractors = s2v.get_best_sense(answer)
        if distractors is not NULL do
            find answer's start and end positions in input context
            mcq_input = "generate question:" + context[ :start_position -
            1] + "<hl>" + context[ start_position : end_position ] +
            "<hl>" + context[ end_position +1 : ]
            generated_mcq_questions = add_to_list( GenerateMcqQuestions(
            mcq_input))
            append generated_mcq_questions, answer and distractors to
            output_dict
        end
    end
    reformat output_dict
End
```

The system automatically generates different types of question-answers over a given text. Figure 6 shows the generated question–answer pairs using the above-mentioned algorithms (Subjective Question–Answer Generation, Boolean Answer–Question Generation, Multiple-Choice Question–Answer Generation, and Fill-in-the-Blanks-type Question–Answer Generation). These algorithms individually generate more than one relevant question–answer pair over the same context.

4 Evaluation

The proposed system is evaluated to analyze the quality of generated question–answers. The performance of the system is evaluated on various datasets and metrics as mentioned below.

4.1 Dataset Used

The proposed model has been trained on three datasets, i.e., SQuAD [42], QuAC [43], and BoolQ [44] to generate different types of objective and subjective question-answers. Table 1 lists the statistics for these datasets. These datasets have been pre-processing by removing the questions which do not have any answers. For the objective question–answer generation, SQuAD and BoolQ datasets are used. And for subjective question–answer generation, QuAC and SQuAD datasets are used.

4.2 Metrics Used

The performance of the question–answer generation system is assessed using the metrics listed below.

- *ROUGE-L* [45] measures recall that by how many words the predicted and reference sentences are similar using longest common subsequence-based statistics.
- *BLEU* [46] measures precision that scores word similarity between candidate and reference sentences. BLEU-1, BLEU-2, BLEU-3, and BLEU-4, use 1-g to 4-g, respectively, for calculation.
- *METEOR* [47] is based on the harmonic mean of recall and precision, with recall weighted higher than precision.
- *F1 Score* [42] measures the average overlap between the ground truth and the predicted answer.

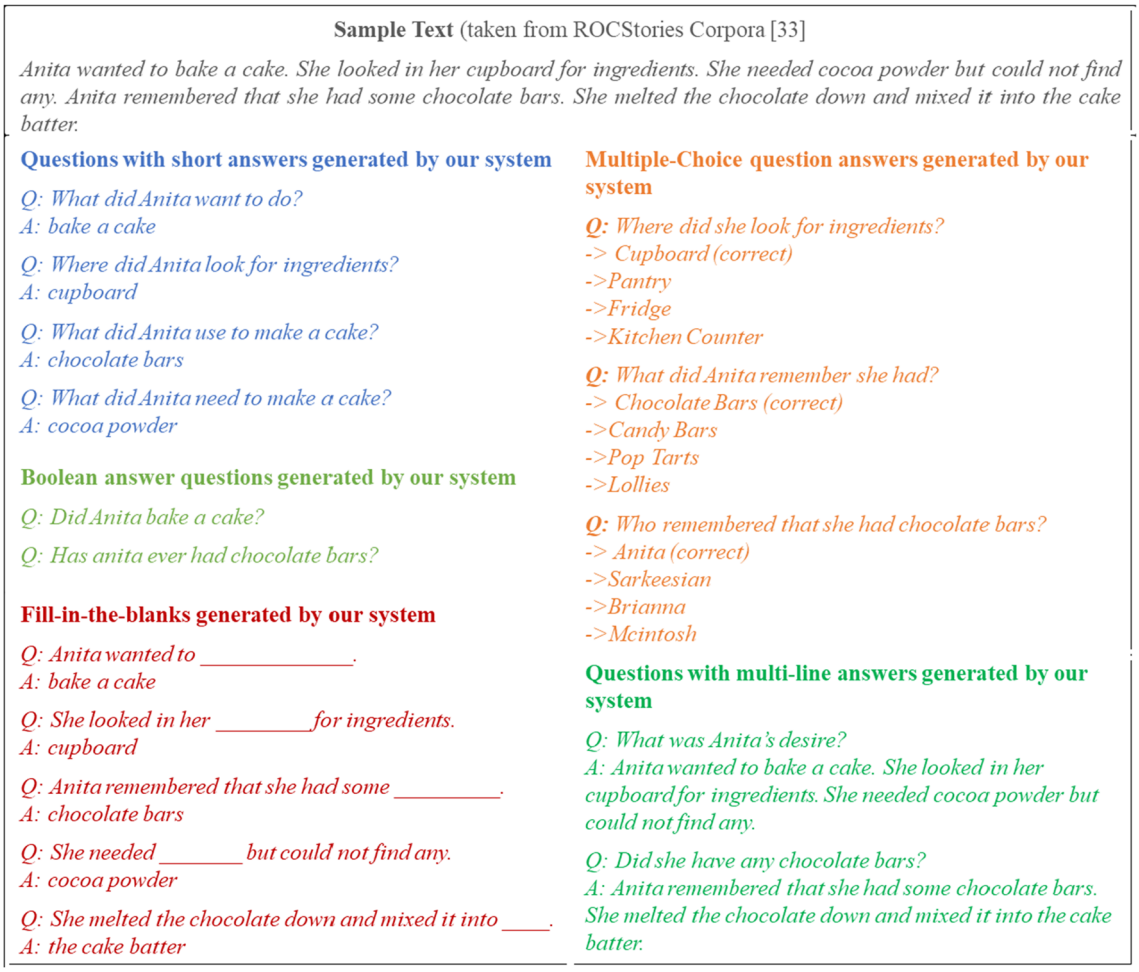


Fig. 6 Sample, diverse question–answer pairs generated by the system over the same given text: (1) questions with short answers, (2) Boolean answer–questions, (3) fill-in-the-blanks, (4) multiple-choice answer–questions, and (5) questions with multi-line answers

4.3 Evaluation Results

The model is evaluated on SQuAD, QuAC, and BoolQ datasets over the above-mentioned metrics. Table 2 lists the metrics and corresponding scores on different datasets.

Table 1 Statistics of the datasets

	SQuAD	QuAC	BoolQ
#Training	59,819	60,674	9427
#Validation	3127	5142	3270
#Test	3000	5140	3270
Average length of context (in words)	121	396	108 tokens
Average length of question (in words)	10	7	8.9 tokens
Average length of answer (in words)	3	14	Boolean

Table 2 Model evaluation scores on different datasets

Dataset	Evaluation metric	Evaluation score
SQuAD	ROUGE-L	40.64
	BLEU-1	41.74
	BLEU-2	30.81
	BLEU-3	23.81
	BLEU-4	18.87
	METEOR	25.24
QuAC	F1	62.71
BoolQ	Accuracy	64.80

The metric scores reflect how similar the generated questions are to the ones in the training dataset and through persistent and careful observation it has been found that almost every time the question and answers formed by the models are grammatically sound and they successfully capture inherent contexts in the given paragraphs.

Table 3 Evaluation of our model with existing approaches

Model	Metrics	
	BLEU-4	METEOR
Du and Cardie [23]	15.16	19.12
Lopez et al. [34]	8.26	21.2
Nema et al. [32]	16.99	21.10
Zhao et al. [31]	16.85	20.62
Kim et al. [33]	16.17	–
Liu [35]	13.86	–
Proposed approach	18.87	25.24

4.4 Evaluation Analysis with Existing Models

The proposed automatically question-answers generation system has been analyzed with the existing models. The evaluation results of the proposed system and other existing models are shown in Table 3. It has been found that the proposed system outperforms on BLEU-4 and METEOR metrics.

It can be seen clearly from the above table that the proposed approach outperformed the existing approaches over BLEU-4 and METEOR metrics. The results mentioned in the above table are the results that the authors report in their respective papers. These papers have utilized different approaches for question–answer generation tasks but are limited in capabilities compared to our proposed approach. Du and Cardie [23] used gated coreference knowledge for paragraph-level answer-aware question generation. Zhao et al. [31] proposed a seq2seq network having a gated self-attention encoder and a maxout pointer decoder for paragraph-level single question generation but lacks diversity. Nema et al. [32] had proposed refine networks for the question generation task and Kim et al. [33] have used deep neural networks to generate questions using answer-separation but are not capable of generating corresponding answers over the same context. Lopez et al. [34] have used a decoder-only transformer for paragraph-level question generation. Liu [35] has used seq2seq-based algorithms with copy and attention mechanisms for question generation. However, the proposed approach used in our paper has utilized T5 transformer architecture to generate question–answer pairs of the selected type such that the questions and answers generated are grammatically and contextually correct. The proposed system has the capability to generate multiple questions–answer pairs over the same sentence/paragraph-level context and the generated answer matches the question in the textual context. The proposed approach has the capability to generate a variety of question–answer pairs, including subjective question-answers, Boolean (yes/no, true/false), fill-in-the-blanks, and multiple-choice question–answer pairs. The generated question–answers are comparable to the baseline question–answer generation models and have performed well with the majority of question–answer domains as shown in Table 4.

It has been observed that the proposed system outperforms the baseline question and answer generation systems. The question and answer pairs generated by our system show diversity over same given textual passage. The generated question–answer pairs have also been shown to a small group of individuals and majority of the questions were found grammatically sound and acceptable over the given context.

5 Conclusion and Future Scope

In this paper, the use of encoder–decoder architecture-based text-to-text transfer transformer (T5) has been introduced to automatically generate sentence and paragraph-level question–answer pairs with diverse perspectives over a given context. The system outperforms the existing baseline question–answer generation models over BLEU-4 and METEOR evaluation metrics with a score of 18.87 and 25.24, respectively. The paper also shows that the proposed system efficiently generates question–answer pairs over the context where the baseline approaches failed. It is believed that the question–answer generation system makes the automatic process of tests creation easier and helps students with their self-evaluation and tutors for academic purposes. Question-answers impromptu class discussions and hence help reinforce the concepts of their students. Also, such an application enables students to test their understanding whenever required, helping them evaluate themselves as well as eradicating the fear generally associated with tests and quizzes.

The system addresses the need for variations of tests as the system offers the users to choose a variety of question–answer types, including short and long answer-type questions, Boolean answer-type questions, multiple-choice question–answer generation, and fill-in-the-blanks-type question–answer generation. The system has the capability to generate question–answer pairs over single text and on a paragraph. The application helps the instructors to save their time by providing them with an easy, efficient, and reliable source to generate questions with corresponding answers. The saved time can then be used to focus on other important academic activities, personal care, and well-being. The practice of questions helps boost the morale of students and helps them gain confidence in their field of study.

Further, it is likely to extend the work including more local Indian languages like Hindi and Punjabi. The more powerful language models like T5-large, T5-3B, and T5-11B would be

Table 4 Analysis of generated question-answers by existing models and by our system with three examples

Example 1
Text 1: There is, I think, humor here which does not translate well from English into sanity

Question–Answer generated by Lovenia, Limanta, and Gunawan 2018 [40]	Question–Answer generated by our system
Q: What does not do?	Q: What does not translate well from English into sanity?
A: sanity	Answer: humor
	Q: What does not translate well into sanity?
	– > Humor (correct)
	– > Dark Humour
	– > Slapstick
	– > Goofiness
	Q: What does humor translate well from English into?
	– > Sanity (correct)
	– > Own Sake
	– > Life
	– > Mental Stability
	Q: There is, I think, _____ here which does not translate well from English into sanity
	Answer: humor

Example 2
Text 2: Liberated by Napoleon's army in 1806, Warsaw was made the capital of the newly created Duchy of Warsaw

Question–Answer generated by Nema et al. 2019 [32]	Question–Answer generated by our system
Q: Who liberated Warsaw in 1806?	Q: When was Warsaw liberated?
	Answer: 1806
	Q: Who liberated Warsaw in 1806?
	– > Napolean (correct)
	– > Genghis
	– > Military Genius
	– > Augustus
	Q: Is Warsaw the capital of the new Duchy ?
	Q: What was the capital of the newly created Duchy of Warsaw?
	– > Warsaw (correct)
	– > Berlin
	– > Moscow
	– > Vilnius
	Q: What was the name of the newly created city of Warsaw?
	– > Duchy (correct)
	– > Duchies
	– > Kingdom Title
	– > Vassal

Example 3
Text 3: Teaching may be carried out informally, within the family, which is called homeschooling, or in the wider community. Formal teaching may be carried out by paid professionals. Such professionals enjoy a status in some societies on a par with physicians, lawyers, engineers, and accountants (Chartered or CPA)

Question–Answer generated by Lopez et al. 2020 [34]	Question–Answer generated by our system
Q: What is a profession of the profession of the profession of the profession of the profession of the profession of the profession of the profession of the profession of the profession	Q: What is the name of the family that may taught informally?
	Answer: homeschooling
	Q: Teaching may be carried out informally, within the family, which is called _____
	Answer: homeschooling
	Q: Who may be responsible for the formal teaching?
	– > Paid Professionals (correct)
	– > Entertainers
	– > Amateurs
	– > Decision Makers
	Q: Formal teaching may be carried out by _____
	Answer: paid professionals
	Q: What is usually carried out informally?
	– > Teaching (correct)
	– > Teach
	– > Learning
	– > College Course

experimented near future. The automation of question–answer generation algorithms would be made faster with more depth and diversity in the future. This paper is believed to benefit the users with the proposed question and answer generation system.

Funding The authors have no funding to report.

Data Availability The data used to support the findings of this study are included within the article.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Consent for publication This work is original and has not been published elsewhere nor is it currently under consideration for publication elsewhere

Ethical Approval The author declares that this article complies with the ethical standard.

References

1. Agarwal, M.; Mannem, P.: Automatic gap-fill question generation from text Books. In: Proceedings of the Sixth Workshop on Innovative Use of NLP for Building Educational Applications, pp. 56–64 (2011)
2. Kumar, G.; Banchs, R.; D'Haro, L.F.: RevUP: automatic gap-fill question generation from educational texts. Assoc. Comput. Linguist. (2015). https://doi.org/10.3115/v1/w15-0618
3. Baha, T.A.I.T.; Hajji, M.E.L.; Es-Saady, Y.; Fadili, H.: Towards highly adaptive Edu-Chatbot. Procedia Comput. Sci. **198**, 397–403 (2021). https://doi.org/10.1016/j.procs.2021.12.260
4. Gao, S.; Ren, Z.; Zhao, Y.; Zhao, D.; Yin, D.; Yan, R.: Product-aware answer generation in E-commerce question-answering. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, pp. 429–437. ACM, New York, NY (2019)
5. Shen, S.; Li, Y.; Du, N.; Wu, X.; Xie, Y.; Ge, S.; Yang, T.; Wang, K.; Liang, X.; Fan, W.: On the generation of medical question-answer pairs. Proc. AAAI Conf. Artif. Intell. **34**, 8822–8829 (2020). https://doi.org/10.1609/aaai.v34i05.6410
6. Liu, S.; Zhang, X.; Zhang, S.; Wang, H.; Zhang, W.: Neural machine reading comprehension: methods and trends. Appl. Sci. **9**, 3698 (2019). https://doi.org/10.3390/app9183698
7. Weston, J.; Chopra, S.; Bordes, A.: Memory networks. In: 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc. pp. 1–15 (2015)
8. Zhou, Q.; Yang, N.; Wei, F.; Tan, C.; Bao, H.; Zhou, M.: Neural question generation from text: A preliminary study. In: Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 10619 LNAI, pp. 662–671 (2018). https://doi.org/10.1007/978-3-319-73618-1_56
9. Kumar, V.; Ramakrishnan, G.; Li, Y.F.: Putting the horse before the cart: a generator-evaluator framework for question generation from text. In: CoNLL 2019 - 23rd Conf. Comput. Nat. Lang. Learn. Proc. Conf. pp. 812–821 (2019). https://doi.org/10.18653/v1/k19-1076
10. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I.: Attention is all you need. In: NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems. CA, USEA. arXiv:1706.03762v5 (2017)
11. Devlin, J.; Chang, M.; Lee, K.; Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805v2 4171–4186 (2019). https://doi.org/10.18653/v1/N19-1423
12. Radford, A.; Narasimhan, K.: Improving language understanding by generative pre-training (2018)
13. Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; Zettlemoyer, L.: BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. 7871–7880 (2020). https://doi.org/10.18653/v1/2020.acl-main.703
14. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. 1–67. arXiv:1910.10683v3 (2020)
15. Chali, Y.; Hasan, S.A.: Towards topic-to-question generation. Comput. Linguist. (2015). https://doi.org/10.1162/COLI
16. Danon, G.; Last, M.: A syntactic approach to domain-specific automatic question generation (2017)
17. Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y.: Learning phrase representations using RNN encoder–decoder for statistical machine translation. Methods Nat. Lang. Process. (EMNLP) Assoc. Comput. Linguist. Empir. (2014). https://doi.org/10.1128/jcm.28.9.2159-.1990
18. Bahdanau, D.; Cho, K.; Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc. pp. 1–15 (2015)
19. Du, X.; Shao, J.; Cardie, C.: Learning to ask: neural question generation for reading comprehension. arXiv:1705.00106v1 (2017)
20. Upadhya, B.A.; Udupa, S.; Kamath, S.S.: Deep neural network models for question classification in community question-answering forums. In: 2019 10th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2019, pp. 6–11. IEEE (2019)
21. Wang, R.; Panju, M.; Gohari, M.: Classification-based RNN machine translation using GRUs. 1–7 (2017)
22. Serban, I.V.; García-Durán, A.; Gulcehre, C.; Ahn, S.; Chandar, S.; Courville, A.; Bengio, Y.: Generating factoid questions with recurrent neural networks: the 30m factoid question-answer corpus. In: Proc. 54th Annu. Meet. Assoc. Comput. Linguist., vol. 1, pp. 588–598 (2016). https://doi.org/10.18653/v1/P16-1056
23. Du, X.; Cardie, C.: Harvesting paragraph-level question-answer pairs from wikipedia. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, vol. 1, pp. 1907–1917. Association for Computational Linguistics, Stroudsburg, PA (2018)
24. Song, L.; Wang, Z.; Hamza, W.; Zhang, Y.; Gildea, D.: Leveraging context information for natural question generation. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 2, pp. 569–574. Association for Computational Linguistics, Stroudsburg, PA (2018)
25. Du, X.; Cardie, C.: Identifying where to focus in reading comprehension for neural question generation. In: Proceedings of the 2017 conference on empirical methods in natural language processing, pp. 2067–2073. Association for Computational Linguistics, Stroudsburg, PA (2017)
26. Sun, X.; Liu, J.; Lyu, Y.; He, W.; Ma, Y.; Wang, S.: Answer-focused and position-aware neural question generation. In: Proceedings of the 2018 conference on empirical methods in natural language

processing, pp. 3930–3939. Association for Computational Linguistics, Stroudsburg, PA (2018)
27. Meng, R.; Zhao, S.; Han, S.; He, D.; Brusilovsky, P.; Chi, Y.: Deep keyphrase generation. In: Proc. 55th Annu. Meet. Assoc. Comput. Linguist., vol. 1, pp. 582–592 (2017). https://doi.org/10.18653/v1/P17-1054
28. Subramanian, S.; Wang, T.; Yuan, X.; Zhang, S.; Trischler, A.; Bengio, Y.: Neural models for key phrase extraction and question generation. In: Proceedings of the Workshop on Machine Reading for Question Answering,. pp. 78–88. Association for Computational Linguistics, Stroudsburg, PA (2018)
29. Willis, A.; Davis, G.; Ruan, S.; Manoharan, L.; Landay, J.; Brunskill, E.: Key phrase extraction for generating educational question-answer pairs. In: Proc. Sixth ACM Conf. Learn. @ Scale. pp. 1–10 (2019). https://doi.org/10.1145/3330430.3333636
30. Liu, B.; Zhao, M.; Niu, D.; Lai, K.; He, Y.; Wei, H.; Xu, Y.: Learning to generate questions by learningwhat not to generate. World Wide Web Conf. - WWW '1, pp. 1106–1118 (2019). https://doi.org/10.1145/3308558.3313737
31. Zhao, Y.; Ni, X.; Ding, Y.; Ke, Q.: Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In: Proc. 2018 Conf. Empir. Methods Nat. Lang. Process. EMNLP 2018, pp. 3901–3910 (2018). https://doi.org/10.18653/v1/d18-1424
32. Nema, P.; Mohankumar, A.K.; Khapra, M.M.; Srinivasan, B.V.; Ravindran, B.: Let's ask again: refine network for automatic question generation. In: Proc. 2019 Conf. Empir. Methods Nat. Lang. Process. 9th Int. Jt. Conf. Nat. Lang. Process. pp. 3312–3321 (2019). https://doi.org/10.18653/v1/D19-1326
33. Kim, Y.; Lee, H.; Shin, J.; Jung, K.: Improving neural question generation using answer separation. In: Proceedings of the AAAI Conference on Artificial Intelligence (2019)
34. Lopez, L.E.; Cruz, D.K.; Cruz, J.C.B.; Cheng, C.: Simplifying Paragraph-level Question Generation via Transformer Language Models. (2020)
35. Liu, B.: Neural question generation based on Seq2Seq. In: Proceedings of the 2020 5th International Conference on Mathematics and Artificial Intelligence, pp. 119–123 (2020). https://doi.org/10.1145/3395260.3395275
36. Akyon, F.C.; Cavusoglu, D.; Cengiz, C.; Altinuc, S.O.; Temizel, A.: Automated question generation and question answering from Turkish texts using text-to-text transformers. 1–14 (2021). https://doi.org/10.3906/elk-Automated
37. Nassiri, K.; Akhloufi, M.: Transformer models used for text-based question answering systems. Appl. Intell. (2022). https://doi.org/10.1007/s10489-022-04052-8
38. Bashath, S.; Perera, N.; Tripathi, S.; Manjang, K.; Dehmer, M.; Streib, F.E.: A data-centric review of deep transfer learning with applications to text data. Inf. Sci. (Ny) **585**, 498–528 (2022). https://doi.org/10.1016/j.ins.2021.11.061
39. Dehghani, M.; Azarbonyad, H.; Kamps, J.; De Rijke, M.: Learning to transform, combine, and reason in open domain question answering. In: CEUR Workshop Proc., vol. 2491, pp. 681–689 (2019). https://doi.org/10.1145/3289600.3291012
40. Lovenia, H.; Limanta, F.; Gunawan, A.: Automatic question-answer pairs generation from text. Acad. Edu. (2018). https://doi.org/10.13140/RG.2.2.33776.92162
41. Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; Davison, J.; Shleifer, S.; von Platen, P.; Ma, C.; Jernite, Y.; Plu, J.; Xu, C.; Le Scao, T.; Gugger, S.; Drame, M.; Lhoest, Q.; Rush, A.: Transformers: state-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 38–45 (2020). https://doi.org/10.18653/v1/2020.emnlp-demos.6
42. Rajpurkar, P.; Zhang, J.; Lopyrev, K.; Liang, P.: SQuAD: 100,000+ questions for machine comprehension of text. In: Proc. 2016 Conf. Empir. Methods Nat. Lang. Process. pp. 2383–2392 (2016). https://doi.org/10.18653/v1/D16-1264
43. Choi, E.; He, H.; Iyyer, M.; Yatskar, M.; Yih, W.; Choi, Y.; Liang, P.; Zettlemoyer, L.: QuAC: question answering in context. In: Proc. 2018 Conf. Empir. Methods Nat. Lang. Process., pp. 2174–2184 (2018). https://doi.org/10.18653/v1/D18-1241
44. Clark, C.; Lee, K.; Chang, M.; Kwiatkowski, T.; Collins, M.; Toutanova, K.: BoolQ: exploring the surprising difficulty of natural yes/no questions. In: Proc. 2019 Conf. North., pp. 2924–2936 (2019). https://doi.org/10.18653/v1/N19-1300
45. Lin, C.-Y.: ROUGE: a package for automatic evaluation of summaries. In: Text Summarization Branches Out. pp. 74–81. Association for Computational Linguistics, Barcelona, Spain (2004)
46. Papineni, K.; Roukos, S.; Ward, T.; Zhu, W.-J.: BLEU: a method for automatic evaluation of machine translation. In: 40th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 311–318. ACL (2002)
47. Lavie, A.; Agarwal, A.: METEOR: an automatic metric for MT evaluation with high levels of correlation with human judgments. In: Proceedings of the Second Workshop on Statistical Machine Translation (2005)