A Strategy for implementing Domain Based Task Generation and Evaluation System Using Text-Text Generative Models

Chukwuka Victor Obionwu[1], Diptesh Mukherjee[1], Vishnu Devadas[1], Shimony Mittal[1], Anam Naimat Ghumman[1], Anjali Katherine John[1], Anja Buch[1], Andreas Nuernberger[1], and Gunter Saake[1]

Otto von Guericke Universität Magdeburg, Germany

Abstract. In both traditional and fully online teaching models, the pedagogical metric for evaluating learning engagement is centered around exercise questions that assess the level of knowledge that learners have acquired with respect to a particular learning engagement. While it is impossible to run out of questions, manually generating contextually relevant and unique questions in a declarative programming course based on structured query language is an inevitable challenge. In our particular situation, we developed an automatic question generation system using the text-to-text transfer transformer and a corresponding evaluation system using the bidirectional encoder representations from transformers. To evaluate the usefulness and effectiveness of our implementation, we evaluated our system against the GPT-3.5 performance baseline. The performance of our prototype reaches a precision value of 0.44 and an $F_{\beta=0.5}$ value of 0.42 while the Gpt 3.5 gives us 0.4 and 0.45 respective score, which justifies our strategy of employing large language models for question/answer generation and evaluation.

Keywords: Natural language processing· Large language models· Web classroom applications· Managed Learning Environments · Technology-Enhanced Learning.

1 Introduction

The increasing use of asynchronous distance learning platforms [12, 13] and recent innovations in the natural language processing research area have led to the gradual but increasing adoption of automated systems in traditional contexts. According to Wang et al, [23], automated question generation and answer evaluation systems (AQGAES) enhance learning experience by automatically generating relevant questions and delivering timely feedback, which improves the students' learning interaction, knowledge acquisition and alleviate the burden of creating questions and evaluating assessments manually. The major innovation

at the center of this automation is the transformer. Transformers are deep learning architectures that utilize an attention strategy to selectively concentrate on specific segments of an input text it predicts to be the most relevant instead of a recurrent neural network [22]. Current research trends indicate that transformer-based pre-trained models can accomplish state-of-the-art performance on a variety of tasks, including speech processing and machine translation [15]. As a result, Transformer has become the architecture of choice in NLP, particularly for applications that combine different modes of communication, e.g., in speech-to-text translation systems [4], question-answering systems [**?**], etc. In our peculiar scenario, we needed to alleviate the administrative burden on instructors while composing exercise tasks and general lecture administration due to the increasing number of students enrolled in our course. In addition, there is the difficulty of composing pedagogically pertinent and contextually distinct tasks that assist students in assessing the level of knowledge they have acquired during their lecture participation. Furthermore, given the significant advantages of automated assessments, including scalability and consistency in evaluating student responses, large-language NLP models can rapidly evaluate a large number of responses while consistently applying the evaluation criteria to each response. This scalability ensures that assessments can be conducted efficiently, even in cases where the number of students or assessments is high. It also potentially addresses the challenges ensuing from subjective grading processes, which are prone to human biases. Thus, in this work, we describe some of our ongoing efforts aimed at generating and evaluating tasks using the Text-to-Text Transfer Transformer (T5) and the Bidirectional Encoder Representations from Transformers (BERT). Further, we aim to shed light on the following research questions:

- How can question generation models be used to support formative evaluations, delivering timely feedback to improve students' learning and growth in university courses?
- How can automatic question generation models based on transformers (such as T5 and BERT) be optimized to generate high-quality, contextually relevant questions that align with the specific learning objectives and topics covered?

The first question is answered on Answer Evaluation Pipeline section of Implementation, whereas the second question is answered on the Question-Answer Pair Generation Pipeline section of Implementation.

The rest of the paper proceeds as follows: The next section deals with the background of our strategies. Section 3 deals with the implementation of our system, and Section 4 provides insight into our evaluation and results. In Section 5, detail the state-of-the-art studies that are related to our effort. We summarize our contributions in Section 6 and indicate directions for future efforts.

2 Background for Our Strategy

Automated question generation is a natural language processing (NLP) task in which questions are automatically generated based on input data such as text

documents, paragraphs, or sentences, using a transformer-based model such as T5 (Text-to-Text Transfer Transformer model) [11]. The T5 model utilizes a sequence-to-sequence approach for learning by taking in a sequence of input text (the context) and producing a sequence of output text (the generated questions) [17]. The core idea behind T5 is that it handle every NLP task as a "text-in, text-out" problem. To gain insight into the T5 language model, we describe the transformer.

Transformer is a neural network architecture that is capable of processing sequential data, including texts, audios, videos, and images, without recurrent or convolutional layers. Its fundamental layer is Attention, and it consists of fully connected, normalization, embedding, and positional encoding layers [22, 6, 3]. Originally, it was intended for neural machine translation, where a source sentence is first encoded into a fixed-length vector, after which a decoder then outputs a translation from the encoded vector. However, in scenarios where the input text is long, the computational cost of using the fixed-length internal representations for the capture of the semantic details of the input text becomes high. Attention is employed in addressing this challenge by allowing the neural network to focus on the sections of input data that contain meaningful information and pay less attention to the rest of the input. Given an encoder-decoder architecture with attention, several innovative strategies, and billions of parameters, we get large language models that are capable of executing natural language downstream tasks via zero-shot learning.

	Layers	Width	Heads	Parameters
BERT-Large	24	1024	16	340 Million
RoBERTa	24	1024	16	355 Million
Turing-NLG	78	4256	28	17 Billion
LaMDA	64	8192	128	137 Billion
GPT-3	96	12,228	96	175 Billion
PaLM	118	18,432	48	540 Billion

Table 1. Large Language Models

In Table 1, we list some of the main stream large language models. In the table, the term "layers" refers to the quantity of encoder-decoder models that are layered on top of each other. The parameter "width" denotes the dimension of the model. Additionally, the parameter "heads" represents the number of attention layers in the multi-head attention mechanism. Lastly, the term "parameters" signifies the total count of parameters utilized by the model.

In an educational setting, these models can compare the answers of each student to the answer key, which is a list of the correct answers. This is done by using semantic textual similarity scoring, which creates a similarity score that

shows how similar the student's answer is to the answer key in terms of meaning and context. A higher similarity score indicates a greater level of correctness with respect to the student's answer. Automated evaluation with models trained for semantic textual similarity can handle diverse phrasings and synonyms. Therefore, it can recognize a wide range of correct answers, provided they convey the same meaning as the expected answer. While MCQs provide a binary output—ccorrect or incorrect—sshort-answer questions evaluated with tools like STS-BERT can provide a continuous score based on the degree of semantic similarity to the correct answer. This nuanced scoring allows for a more precise assessment of a student's understanding. In the context of our methodology, in this paper, we focus on short answer questions since many of the studies conducted have proven that short answer exercises can enhance students long-term memory, thereby improving their learning performance.[21]. In the next section, we describe our implementation.

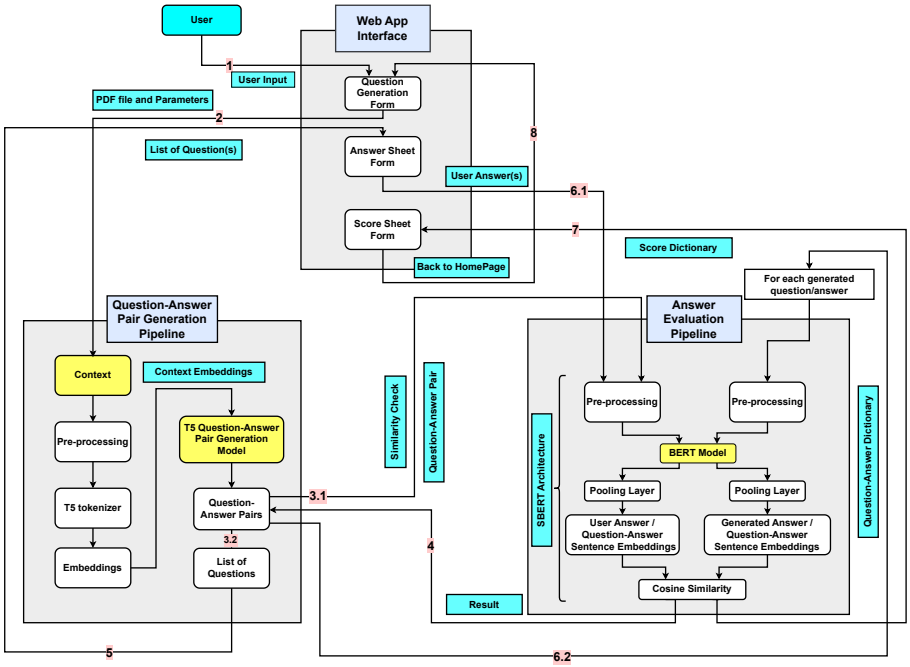

Fig. 1. System Architecture

3 Implementation

Our strategy is to utilize a Question-Answer Pair Generation pipeline (using the t5-small model) and an Answer Evaluation pipeline (using the SBERT model),

along with a simple Anvil web app user interface, to act as an assessment tool for the user. The user initiates the process by providing a PDF file and certain parameters based on which the question-answer pair is generated. Subsequently,the questions are provided to the user for answering, following which the user can submit his or her response in the form of a short answer(s) for further evaluation. The evaluation of the answers is done by comparing the answer given by the user for each generated question to a previously generated answer to the corresponding question. This comparison is done based on semantic similarity using the answer-evaluation model, and a score is assigned to each answer given by the user. This score is then sent to the user through the UI, where he can see the individual score for each question and the overall grade of his performance. This process can be repeated any number of times with any other PDF document of our choice with either the same or different parametric values.

3.1 System Schematic

The system schematic is depicted in Fig 1.

The following steps explain the workflow of the whole system.

1. User inputs the required data (PDF and parameters) in the question generation form of the web app and submits through the provided submit button.
2. The user provided PDF and parameters are sent to the Question-Answer pair generation pipeline.
3. The generated question-answer pair and the question-answer pair dictionary is sent to the Answer evaluation pipeline.
4. After similarity check, the result is provided to the Question-Answer pair generation pipeline and then stored into the dictionary.
5. Now the generated question list is sent to the answer sheet form so that the user can answer.
6. After the user submits the answer(s), the answer(s) and the generated dictionary is sent to the Answer evaluation pipeline for further processing.
7. Now a score dictionary is generated, which contains question as key and score as value. This is sent to the score sheet form so that user can assess how well he performed through score for each question and the normalized final score.
8. After viewing the score sheet form the user can navigate back to the Homepage using the Home button provided in the UI.

In the next subsection, we elaborate on the Question-Answer Pair Generation Pipeline

3.2 Question-Answer Pair Generation Pipeline

Using the pre-trained T5 model, we are generating question-answer pairs, and these are saved into a Python dictionary with the question as the key and the corresponding answer(s) as the value. The format of the dictionary is {Question

: [Answer]}. The process of question generation starts when the user interacts with the pipeline through the web app interface. The step-by-step description of the process is as follows:

- **User Input:** The user inputs (PDF file and parameters) are provided to the pipeline.
- **Context:** According to the given parameters, the context is extracted from the PDF.
- **Pre-processing:** Standard preprocessing of the extracted context is carried out.
- **T5 tokenizer:** To convert the pre-processed data into a format suitable for processing by the T5 model.
- **Embeddings:** Representing individual words as dense vectors, where each element in the vector represents a feature or property of the word.
- **Model:** We are using a t5-small model, which was then fine-tuned on QuAC, and a generated dataset based on scientific research papers.
- **Question-Answer Pair Generation:** Using the above model, the question-answer pairs are generated and these are stored in a Python dictionary for easy access.
- **List of questions:** The list of questions is generated using the keys of the dictionary. This list is then provided to the user through the UI, where the user can attempt to answer the questions and then submit them for evaluation.

The QA Generation algorithm extracts questions and answers from PDF, removing figure and table content as a preprocessing step for the PDF document. It generates QA pairs by processing paragraphs on each page as illustrated in Table 2 & 3 while staying within specified QA limits per page. The Answer Evaluation pipeline is also utilized here in the background to check whether a question that is similar to the currently generated question exists in the aforementioned dictionary. This approach results in the generation of diverse questions and a corresponding possible list of answers for each of the questions generated, which is discussed in detail in the Answer Evaluation pipeline. The process is further illustrated in Algorithm 1.

Details regarding the pipeline components and functionalities are as follows:

- **Transformer model:** T5-small
- **Dataset used for Fine-tuning:** QuAC and Generated Dataset
- **Process:** Question-Answer Pair generation
- **Input:** PDF file, Number of questions to be generated, the start and end page number from which the questions are to be generated.
- **Output:** Question-Answer pair from which we are generating

4 Evaluation and Result

In this section, we provide the datasets used and the evaluation metrics, along with the results of the performance evaluation for both the Question-Answer Pair Generation task and the Answer Evaluation task.

Algorithm 1 QA Generation Algorithm

Require: pdf_document, starting_page, ending_page, num_qa_required, max_qa_per_page
Ensure: Question_Answer_Dictionary
Initialize $QA\_Dictionary = \{\}$
Define $figure\_regex, table\_regex$
Calculate $num\_qa\_per\_page\_ceil = \lceil \frac{num\_qa\_required}{ending\_page - starting\_page + 1} \rceil$
if $num\_qa\_per\_page\_ceil > max\_qa\_per\_page$ **then**
    Print "Enter_fewer_questions"
else
    Read $pdf\_document$
    for $page$ in $starting\_page$ to $ending\_page + 1$ **do**
        $page = read(page)$
        $page = apply\_regex(page, figure\_regex, table\_regex)$
        $paragraphs = gen\_paragraphs(page, num\_qa\_per\_page)$
        for $paragraph$ in $paragraphs$ **do**
            $question, answer = extract\_qa(paragraph)$
            if $count < num\_qa\_required$ **then**
                $generated\_ques, generated\_ans = QuesAnsModel(t5 - small)$
                if $len(QA\_Dictionary) = 0$ **then**
                    $QA\_Dictionary[generated\_ques] = [generated\_ans]$
                    $count = count + 1$
                end if
                if $count > 0$ **then**
                    $similar\_ques = Process\_Q(generated\_ques, QA\_Dictionary.keys())$
                    if $similar\_ques =$ "" **then**
                        $QA\_Dictionary[generated\_ques] = [generated\_ans]$
                        $count = count + 1$
                    else
                        $answer\_list = QA\_Dictionary[similar\_ques]$
                        $similar\_ans = Process\_A(generated\_ans, answer\_list)$
                        if $similar\_ans = False$ **then**
                            $answer\_list.insert(generated\_ans)$
                            $QA\_Dictionary[generated\_ques] = answer\_list$
                        end if
                    end if
                end if
            end if
        end for
    end for
    Return $QA\_Dictionary$
end if

Input Context to the Model
We present QuAC, a dataset for Question Answering in Context that contains 14K information-seeking QA dialogs (100K questions in total). The dialogs involve two crowd workers: (1) a student who poses a sequence of freeform questions to learn as much as possible about a hidden Wikipedia text, and (2) a teacher who answers the questions by providing short excerpts from the text. QuAC introduces challenges not found in existing machine comprehension datasets: its questions are often more open-ended, unanswerable, or only meaningful within the dialog context, as we show in a detailed qualitative evaluation. We also report results for a number of reference models, including a recently state-of-the-art reading comprehension architecture extended to model dialog context. Our best model underperforms humans by 20 F1, suggesting that there is significant room for future work on this data. Dataset, baseline, and leaderboard available at http://quac.ai

Table 2. Model's Input

Sample Questions & Answers Generated from the Models	
Proposed Model	**GPT-3.5 Turbo**
Question 1: What are the challenges presented in the QuAC dataset? Answer: The challenges encountered in existing machine comprehension datasets include open-ended, unanswerable, or meaningful within the dialog context, as shown in qualitative evaluation.	Question 1: What challenges does the QuAC dataset introduce that are not present in existing machine comprehension datasets? Answer: The QuAC dataset introduces challenges such as open-ended, unanswerable, or context-dependent questions, which are not commonly found in existing machine comprehension datasets. These questions may only be meaningful within the specific context of the dialog. These challenges were identified through a detailed qualitative evaluation of the dataset.

Table 3. Model's Output

4.1 Dataset Used

The proposed Question-Answer Pair Generation model has been trained on QUAC [1] and a custom-generated dataset. The motivation to use these datasets was to focus on abstractive question answers and scientific text. As there were no specific datasets available for scientific text, we created a custom-generated dataset. This dataset was manually generated by us and consists of context and question-answer pairs created from the relevant research papers gathered during the literature survey. All these papers are mentioned in the references.

The proposed answer evaluation model has been trained on the STSB and SICK datasets. These data sets cover a wide range of domains and types of texts, which are well-established sources for evaluating the performance of STS models.

4.2 Evaluation Metrics

The performance of the question-answer pair generation model is assessed using the metrics listed below.

– BLEU [14] measures the precision that scores word similarity between candidate and reference sentence.

– ROUGE-1 [9] refers to the overlap of unigrams (each word) between the system and reference summaries.
– METEOR [8] is based on the harmonic mean of recall and precision, with recall weighted higher than precision.
– MSE-Loss Function measures the average squared difference between the predicted similarity scores and the ground truth similarity scores.

4.3 Evaluation Results

In our assessment of the Question-Answer Pair Generation task, we embarked on a comprehensive comparison involving our model and the GPT 3.5 Turbo variant. These models underwent meticulous evaluation against human-crafted answers, utilizing well-established word-overlap metrics. Subsequently, we closely scrutinized and compared the performance of both models.

Our evaluation process entailed subjecting the two models to a set of five selected questions, drawn from a specific contextual domain. The resulting insights from Fig4.3 reveal a distinct pattern. Except for questions 1 and 5, GPT consistently outperforms our model. This performance disparity is notably attributed to the sheer parameter magnitude of GPT, boasting 175 billion parameters compared to our model's configuration with 64 million parameters in the T5-small setup.

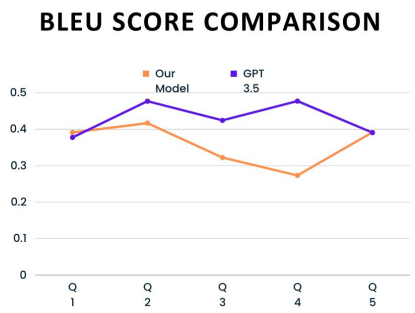

Fig. 2. BLUE Score comparison

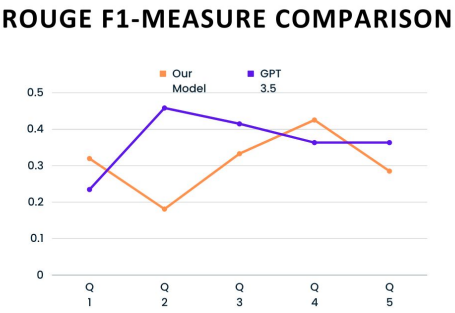

Fig. 3. ROUGE F1-Measure Comparison

Notably, in terms of precision as discerned through ROUGE scores in Fig4, our model stands out as a superior contender. It excels in three out of the five instances evaluated, showcasing its prowess in generating precise and accurate responses. Conversely, GPT consistently achieves remarkable recall across all instances, signifying its strength in comprehensive context recall. And the F1 score depicted in Fig4.3, except for one instance GPT is outperforming our model.

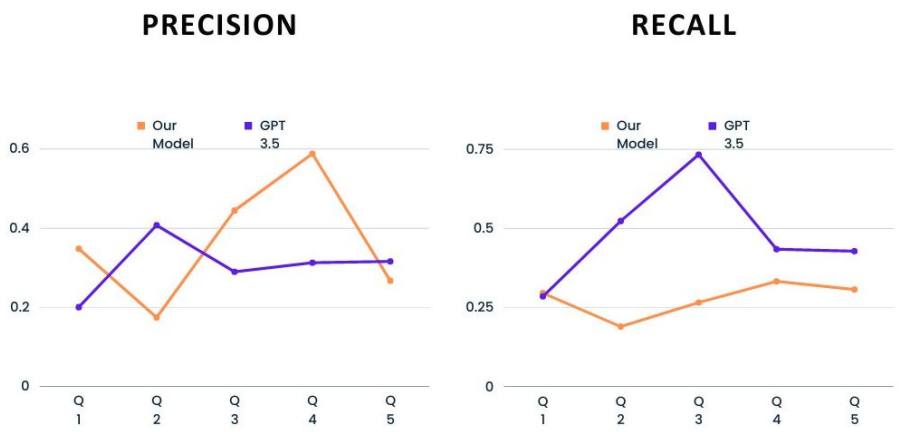

Fig. 4. Precision-Recall ROUGE score comparison

Delving deeper into the results, Fig4.3, provides additional insights into our model's proficiency. The graph illustrates our model's commendable performance in 60% of the cases analyzed.

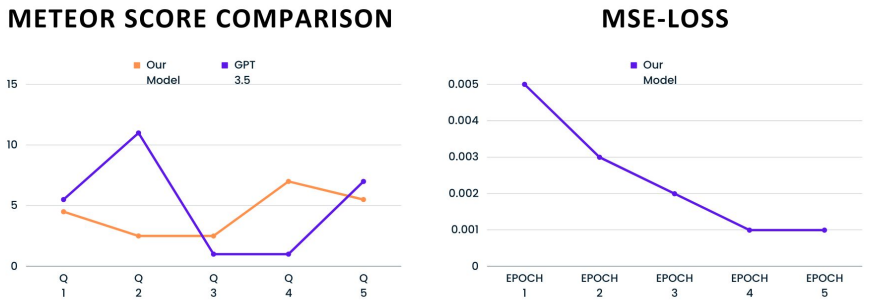

Fig. 5. METEOR Score Comparison

Fig. 6. MSE-LOSS Curve

We have used MSE Loss to quantify the discrepancy between the predicted similarity scores and the ground truth similarity scores in the answer evaluation task. This measure has been considered for evaluation as it is particularly well suited to continuous-valued similarity scores since it penalizes larger differences between predicted and actual scores more severely. Fig4.3 presents the MSE Loss values obtained for our model after each successive epoch, on the concatenated dataset comprising both STSB and SICK datasets. The low MSE Loss values in-

dicate the model's effectiveness in capturing semantic textual similarity, thereby showcasing its capability of accurately measuring semantic relationships between sentences.

5 Related Works

Several studies have been conducted in the past to explore the evaluation of text-to-text generative models with the aim of enhancing learning in educational contexts. In the following, we review related works on question generation and evaluation systems. In rule-based approaches for question answer generation systems, human specialists develop rules or templates to transform a given input text into a set of questions [5]. The primary procedure entails preprocessing a given text in order to select specific answers for which questions can be generated based on a specified template. The most difficult aspect of this strategy is the need for a domain-specific human expert. In contrast to conventional approaches that rely heavily on fixed heuristic rules for converting sentences into associated questions, neural question generation models utilize the encoder-decoder architecture and attention mechanism to produce a wide range of meaningful questions from natural language sentences [2][19]. This strategy further involves the utilization of various methodologies to integrate the answer information into the generation model. One approach involves utilizing an answer position indicator [10][24], while another approach involves employing an encoding mechanism for the answers [7]. Additionally, there exists a method that involves embedding the relative distance between the context words and the answer [20]. Nevertheless, even when considering context and answer information as input, the issue of question generation remains the problem of mapping one input to multiple outputs. The utilization of large language models for question and answer generation is demonstrated in [18] [16]. In their study, Radford et al. [16], demonstrated the capabilities of OpenAI Codex in generating programming exercises, complete with sample solutions and test cases, as well as providing code explanations. The assessment of these outputs was conducted using both qualitative and quantitative methods. The findings of their study indicate that a significant portion of the content generated through automated means exhibits characteristics of novelty and coherence. Furthermore, in certain instances, the generated questions were deemed suitable for immediate utilization without any further modifications. During the process of exercise creation, it was observed that the programming concepts and contextual themes embedded within the exercises can be significantly influenced by the provision of keywords as input to the model. Even though there still needs to be some kind of oversight to make sure that the generated content is educationally sound before it is given to students, the analysis shows that there is a lot of value in using large-scale generative machine learning models to come up with question-and-answer pairs. Compared to the above research efforts, the research direction taken by Radford et al. [16] is in line with our objective.

6 Summary and Future work

In this work, we described our ongoing efforts at generating and evaluating tasks using the Text-to-Text Transfer Transformer (T5) and SBERT. Our system provides a robust and scalable solution for efficient question generation, evaluation, and feedback. The combination of SBERT architecture and T5 proved beneficial in the automation of assessments, which is scalable and effective for educational settings. Our approach significantly reduces manual effort to generate and evaluate questions, providing students with objective and timely feedback.

Our future work consists of optimizing the model performance using more extensive context-specific datasets as well as incorporating larger variants of the t5 model. Currently, the model works in a context-specific manner, adhering to one particular domain (scientific text), however, this can be extended to various other domains (for example, SQL) by making use of transfer learning.

References

1. Choi, E., He, H., Iyyer, M., Yatskar, M., Yih, W.t., Choi, Y., Liang, P., Zettlemoyer, L.: Quac: Question answering in context. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. pp. 2174–2184 (2018). https://doi.org/10.18653/v1/D18-1241, https://doi.org/10.18653/v1/D18-1241
2. Du, X., Shao, J., Cardie, C.: Learning to ask: Neural question generation for reading comprehension. arXiv preprint arXiv:1705.00106 (2017)
3. Guo, M.H., Xu, T.X., Liu, J.J., Liu, Z.N., Jiang, P.T., Mu, T.J., Zhang, S.H., Martin, R.R., Cheng, M.M., Hu, S.M.: Attention mechanisms in computer vision: A survey. Computational visual media **8**(3), 331–368 (2022)
4. Han, C., Wang, M., Ji, H., Li, L.: Learning shared semantic space for speech-to-text translation. arXiv preprint arXiv:2105.03095 (2021)
5. Heilman, M.: Automatic factual question generation from text. Ph.D. thesis, Carnegie Mellon University (2011)
6. Khan, S., Naseer, M., Hayat, M., Zamir, S.W., Khan, F.S., Shah, M.: Transformers in vision: A survey. ACM computing surveys (CSUR) **54**(10s), 1–41 (2022)
7. Kim, Y., Lee, H., Shin, J., Jung, K.: Improving neural question generation using answer separation. In: Proceedings of the AAAI conference on artificial intelligence. vol. 33, pp. 6602–6609 (2019)
8. 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)
9. Lin, C.Y.: Rouge: A package for automatic evaluation of summaries. In: Text summarization branches out. pp. 74–81 (2004)
10. Liu, B., Zhao, M., Niu, D., Lai, K., He, Y., Wei, H., Xu, Y.: Learning to generate questions by learningwhat not to generate. In: The world wide web conference. pp. 1106–1118 (2019)
11. Nguyen, H.A., Bhat, S., Moore, S., Bier, N., Stamper, J.: Towards generalized methods for automatic question generation in educational domains. In: European conference on technology enhanced learning. pp. 272–284. Springer (2022)
12. Obionwu, V., Broneske, D., Hawlitschek, A., Köppen, V., Saake, G.: Sqlvalidator–an online student playground to learn sql. Datenbank-Spektrum **21**, 73–81 (2021)

13. Obionwu, V., Broneske, D., Saake, G.: A collaborative learning environment using blogs in a learning management system. In: International Conference on Computer Science and Education in Computer Science. pp. 213–232. Springer (2022)
14. 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)
15. Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., Huang, X.: Pre-trained models for natural language processing: A survey. Science China Technological Sciences **63**(10), 1872–1897 (2020)
16. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al.: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)
17. 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. J. Mach. Learn. Res. **21**(1) (jan 2020)
18. Sarsa, S., Denny, P., Hellas, A., Leinonen, J.: Automatic generation of programming exercises and code explanations using large language models. In: Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 1. pp. 27–43 (2022)
19. 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. arXiv preprint arXiv:1603.06807 (2016)
20. 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 (2018)
21. Tsai, D., Huang, A., Lu, O., Yang, S.: Automatic question generation for repeated testing to improve student learning outcome. In: Chang, M., Chen, N.S., Sampson, D., Tlili, A. (eds.) Proceedings - IEEE 21st International Conference on Advanced Learning Technologies, ICALT 2021. pp. 339–341. Proceedings - IEEE 21st International Conference on Advanced Learning Technologies, ICALT 2021, Institute of Electrical and Electronics Engineers Inc. (Jul 2021). https://doi.org/10.1109/ICALT52272.2021.00108, publisher Copyright: © 2021 IEEE.; 21st IEEE International Conference on Advanced Learning Technologies, ICALT 2021 ; Conference date: 12-07-2021 Through 15-07-2021
22. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
23. Wang, H.C., Maslim, M., Kan, C.H.: A question–answer generation system for an asynchronous distance learning platform. Education and Information Technologies (Mar 2023). https://doi.org/10.1007/s10639-023-11675-y, https://doi.org/10.1007/s10639-023-11675-y
24. Zhou, Q., Yang, N., Wei, F., Tan, C., Bao, H., Zhou, M.: Neural question generation from text: A preliminary study. In: Natural Language Processing and Chinese Computing: 6th CCF International Conference, NLPCC 2017, Dalian, China, November 8–12, 2017, Proceedings 6. pp. 662–671. Springer (2018)