Proceedings of the Sixth International Conference on Intelligent Computing and Control Systems (ICICCS 2022)
IEEE Xplore Part Number: CFP22K74-ART; ISBN: 978-1-6654-1035-9

2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS) | 978-1-6654-1035-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICICCS53718.2022.9788264

Deep learning based Answering Questions using T5 and Structured Question Generation System'

Atharva Malhar
Department of Information *Technology*
Sardar Patel Institute of Technology
Mumbai, India
atharvadmalhar@gmail.com

Prerana Sawant
Department of Information *Technology*
Sardar Patel Institute of Technology
Mumbai, India
preranas14@gmail.com

Yash Chhadva
Department of Information *Technology*
Sardar Patel Institute of Technology
Mumbai, India
yashchhadva@gmail.com

Swapnali Kurhade
Department of Information *Technology*
Sardar Patel Institute of Technology
Mumbai, India
swapnali.kurhade@spit.ac.in

***Abstract*—Since there is an agile change in all the domains, it is imperative for students to keep themselves updated to gain expertise in their field of study. Although there is easy access to resources from the web, learning only becomes complete after a thorough system of assessment to identify gaps in the learning process if any. Automatic question generation techniques have been introduced to reduce the time associated with the manual construction of questions. Researchers have come up with state-of-art models to generate these questions. Models like BERT and GPT gained popularity among the masses due to their deduction of complex sentences. These have however been replaced with advanced models such as Transformers which are much more versatile. This paper has fine-tuned one such transformer and proposed a custom model to generate questions. Our system mainly focuses on two types of questions, namely multiple-choice, and long answer types. The generator model is primarily concerned with producing questions from selected phrases and generating semantically motivated phrase-level distractors as MCQ response choices. In contrast, the evaluator model is concerned with identifying the most relevant question-answer combinations. This research work has gained the ability to acquire 76% syntactic accuracy and 84% fluency from the question-answer pairs evaluated using our approach. Along with this, the proposed result has achieved a final accuracy of 82% from the evaluator's bert-base-cased model.**

Keywords—Deep Learning, Question Generation, T5 transformer, Wordnet

I. INTRODUCTION

Virtual learning has become a crucial element of our education in recent years. It eliminates the requirement for you to be physically present in a classroom while accessing the most significant resources for learning a skill set. Without a thorough examination of the material, the learning process appears incomplete. It's a method of assessing a student's comprehension of a topic after the class has ended. It not only allows students to practice recovering what they've learned, but it also aids instructors in identifying loopholes and bridging any gaps that may exist.

It costs money and effort to appoint a panel of subject matter experts to ask questions and then analyze the answers. It would be much easier and more efficient if this process could be automated. For simplicity, assessments are frequently classified as objective or subjective. Subjective assessment requires the student to elaborate and express their opinions on a particular question. In contrast, accurate assessment requires the student to color and express their views on one specific question.

Because resolving discrepancies when a computer is complex, we divide our system into two sections: one for single word answers that are then used to create closely related words called distractors that can be used as multiple-choice questions, and the other for long answer questions. Creating questions from simple assertive sentences necessitates contextual awareness and semantic comprehension. We can build generator models that can form semantically and syntactically correct sentences by leveraging our expertise in cutting-edge models such as T5 and optimizing it on larger datasets. We also propose that our custom structured question generation model generate additional question-answer combinations that can be fed into the distractor model. This type of model can be used in the educational area to set and assess questions and in automated systems like online search engines and chatbots.

II. LITERATURE SURVEY

There have been many studies in the past regarding the generation of question-worthy sentences automatically. Most of them focus on the idea of where to focus while generating questions or the idea of selecting the sentences that generate good questions. Saroj et al.[1] considers frequency, centrality of the word, position, and strength of the neighboring nodes to calculate the importance of the word. Text is preprocessed into a graph, with nodes representing tokens. The author assigns weights to the nodes based on the research's criteria.

A hierarchical neural sentence-level sequence tagging model is proposed by Xinya Du and Claire Cardie [2]. A multi-layer neural network is used to select sentences from a text. In this paper, the text is translated into natural questions using an attention-based sequence-to-sequence learning framework

The system implemented by the authors in [3] shows a neural network-based encoder-decoder design that generates questions for random sentences as answers. The model is governed by two sources of data in this case: a text that should

be the subject of the inquiry and an answer that fits the created question.

Rui Portocarrero et al.[4] improves on the existing TextRank solution. The window-based approach makes use of a snapshot of a small chunk of data and only the top-K words are transferred between snapshots. The Incremental algorithm constantly updates the word graph through the use of a window of words. Additionally, the top-K keywords are updated in each set of text streams, allowing the model to operate efficiently on large datasets.

The work of Katira[5] uses both syntactic and semantic parsing of phrases prior to the question generating step because it is difficult to examine text at a syntactic level when dealing with biological processes. The quality of the generated questions is assessed in terms of their correctness in syntax and semantics, as well as their relevance to the given text.

The system implemented by the authors in [6] tends to address concerns with earlier question generation models, such as faulty interrogative word generation and copying irrelevant context for questions. The answer-focused framework depicts a question word generation model that includes a specific vocabulary of interrogative words that are precisely matched to the answer type. The model can identify appropriate text for questions using the position-aware algorithm.

In [7], the author proposed a solution for dealing with the problem of distractor generation in multiple-choice questions. A question and context reformation module that verifies the correctness of the distractors, and a distraction generator that controls the level of plausibility of the formed distractors make up the EDGE (question and answer guided Distractor Generation) architecture. The model takes a passage and a meaningful response as input and examines the context for other keywords that could be used to generate similar but incorrect multiple-choice distractors.

Models created in the past generated questions from all the sentences in the context. It was critical to retrieve relevant terms from the data and use them as possible answers to generate questions. The proposed approach in [8] uses a random forest classifier to evaluate each sentence in the passage and find question worthy sentences by using context-based and sentence-based attributes of the sentence.

Melissa Roemmele et al.[9] addresses the challenge of generating question-answer items for multi-paragraph documents by adapting an existing pipelined approach that applies a retrieval module to select the most relevant paragraphs, and then a reader module for extracting answers from the retrieved paragraphs, to additionally leveraging a pre-trained text encoding model. A shortcoming of this research is the generalizability across datasets. State-of-the-art QA systems have matched human-level performance on individual datasets like SQuAD, but not the case when evaluated on a generic dataset.

In this work[10], the authors investigate several answer selection, question generation, and filtering methods that form a synthetic adversarial data generation pipeline. This takes human-generated adversarial samples and unannotated text to create synthetic question-answer pairs. It is observed that models become considerably more difficult to beat by human adversaries, with a drop in macro-averaged validated model error rate when compared to non-augmented models.

The proposed method in [11] analyzes the questions by summarizing the text contents using the preference learning algorithm which is optimized with the help of the fireflies' algorithm. The sentences in the summary are transformed into stem for the MCQs. The distractors are generated using similarity metrics such as hypernyms and hyponyms. The system also generates analogy questions to test the verbal ability of the students.

The proposed system in [12] selects potential sentences with the help of existing test items on the web. The sentences are selected using a set of patterns extracted from the existing questions. For generating quality named entity distractors certain additional attribute values on the key are extracted from the web and Wikipedia for entities having similar attribute values are searched.

III. PROPOSED SYSTEM

A. System Overview

The process flow, as depicted in Fig. 1, begins with the user's input. Any format, including a text file or a text link, can be used as input. The user can additionally set the number and type of questions to be generated from the given data, such as descriptive, MCQ, or all. The input text is then preprocessed which includes segmentation, lemmatization, and stemming. The text is then summarized using BERT Extractive Summarizer [15] to extract most of the important sentences while also maintaining the structure of the sentences of the original text. Based on the type of question selected by the user different models are included to generate questions.

If the user selects descriptive, the extracted target sentences are passed to a fine-tuned T5 [18] model and a custom SQG model to generate the question-answer pairs, whereas if the question type is MCQ, the answer keyword is passed to a distractor generation model, which uses word sense disambiguation to verify the context of the key. If the word-sense exists the distractors are generated using Wordnet [17] else they are generated using the T5 model. These question-answer pairs are then run through a question-evaluator model, which assigns a score to each pair based on the quality of the question and answer generated. The top N question-answer pairs are then extracted and presented as system output, where N is the number of questions requested by the user.

B. Text Preprocessing

Preprocessing occurs before passing the text for question generation. We extract noun phrases of the sentence using baseline Part-of-Speech tagging. The other features are extracted using the NER labeling model.NER comprises combing through text data for noun phrases, also known as

named entities, and grouping them with labels like "person," "organization," and "brand". These features together form the input to the question generation model. Along with this, the system uses a more holistic candidate selection method based on the keyword distribution in the dataset. Then a term is selected by the model based upon the input passed. In a dependency parse of the paragraph, answers are then removed by the left and right neighbors of the selected term. To reduce the complexity of the term space, the system tokenizes the file into individual words before stemming and lemmatization. Stemming algorithms are used to convert morphological variants into basic words, which improves the efficiency of information retrieval. Here the model would normalize the words based on their vocabulary and morphological inspection to limit the number of words that match the stems accurately, saving time and memory space. HMM, stemmer has been used in the system, followed by a BERT extractive summarizer which is a encoder-decoder based transformer to enhance the quality of sentences and correctly identify critical information, giving the most relevant candidates to consider for question generation.

C. Distractor Generation

To generate distractors, we first comprehend the meaning of the word in the sentence. Using word sense disambiguation [16], our model first determines the word's meaning. So the function get_wordsense tries to get the correct sense of the word given in the sentence. Once the sense of the word is identified, the get_distractors_wordnet function is called to get the distractors. This function tries to get the distractors with the help of hypernyms and hyponyms of the keyword. A hypernym is a term that refers to a larger category of words. A hyponym is a word with a more particular meaning that falls under that category.

Now all the possible hypernyms of the key and the corresponding hyponyms for each hypernym are found. These hyponyms are considered potential distractors. Potential distractors that share the same part of speech structure as the key have a higher scoring than ones that don't. Any three potential distractors with higher ranks are chosen randomly as the final distractors.

If the number of distractors generated from Wordnet is not enough, the model finds words in the text similar to the key used as distractors for the multiple-choice questions. Every word in the text is categorized into different entities using Named Entity Recognition. The words in the text with the same NER label as the answer key are selected as the possible distractors.

D. Question Generation

Structured Question Generation (SQG System):
The flow of the model as depicted in Fig 2 goes as follows:
Before using this model, we need to use tokenizers that are used to break strings down into lists of substrings. Then we use a part-of-speech tagger, which reads the text in a language and assigns parts of speech to each word, such as nouns, verbs, adjectives, and so on. Depending on the conjunctive markers, sentences are classified into two types - Interrogative and Assertive based sentences.

1. Interrogative Based Templates

This part of the code is triggered if the sentence contains any conjunctive words like 'because', 'for example' etc. After that, we decide which part of the sentence will be used to generate the question. For every conjunctive marker, we have assigned an appropriate question type. This is followed by selecting the sentences with auxiliary verbs and placing the verb at the start of the sentence followed by the rest of the sentence. Now if a verb is not present in the sentence, then we need to add it at the start of the sentence using a custom pairing that we design using the tags that we get during preprocessing. Below is an example of the type of pairing that we have done for present tense sentences.

Pronouns	Verbs
He, She, It	Does
They, Them	Do

Table 1. Pronouns and corresponding Verbs

2. Assertive Based Templates

This category includes assertive or declarative sentences that lack a conjunctive marker. Here again, since we do not have verbs to use at the start of a sentence, we have to use the tags that we recovered during pre-processing of text. Along with Part-of-Speech tags we also use chunk tags for identifying the role of a word in the sentence. Similar to what we did earlier, we use the verb recovered from the custom pairing using tags and using this verb at the start of a sentence. For imperative sentences, we can simply put a question mark at the end since these are binary questions.

We employ another pairing rule base in this section. For every named entity, we assign an appropriate question tag. For example, time is associated with 'When?' and objects are associated with 'What?'. If such entities occur at the start of a sentence, then we simply replace this with the paired question, else we need to add all the non-auxiliary verbs to the sentence followed by adding the paired question tag at the start of the sentence.

Entity	Question Tag
Time	When
Person	Who
Organization/Event/Object	What
Place	Where

Table 2. Entity Question Tag Pairs

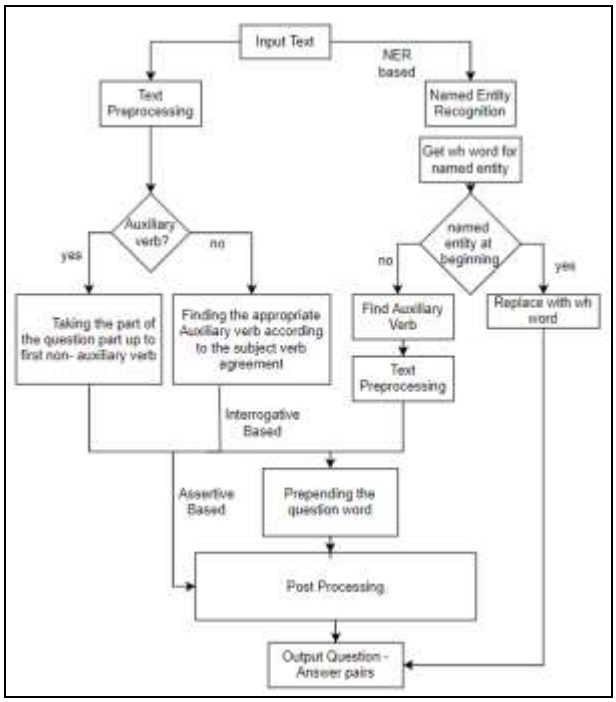

Fig. 2. SQG Model

I. Extractive Question Generation

Once we receive the text after preprocessing, the text is passed down to our custom SQG model and the T5 transformer that we fine-tuned for our use case. T5 is basically an encoder-decoder model that proposes handling all NLP tasks under one umbrella text-to-text format. The T5 model is also trained through the same masked label modeling. It has been trained on the C4 dataset which is essentially an extracted processed text from the internet.

While working with T5, it becomes important to gather good quality datasets which would in turn produce better performance while testing. For training, we used a combination of rows from SQuad[13] and CoQA[14] which made up for the majority of the dataset containing about seventy thousand rows. Next, we needed to add tokens in our dataset to put our data in a structured format to give as inputs in the model. The model's inputs were - questions, responses, and the context around these questions. To tokenize the words, we used the T5 tokenizer which is significantly faster than the BERT. The sentences are prepended by context token (<context>) and answer token (<answer>) We maintained a maximum sequence length of 512 and used optimized stochastic gradient descent. The model is fed a particular token for the responses and context for it to identify the components of the input sequence.

Here we have used GEGLU activation in the hidden layer. The number of buckets used for each attention layer was 64, the dropout rate was set to 0.8 and the maximum distance of long sequences was set to 256 which gave us the best performing model. The encoder model contains two sets of self-attention combined with the normalization layer. The decoder model contains a self-attention layer followed by normalization and encoder-decoder attention. The training model's inputs are the questions, responses, and the context around those questions.

We used a combination of beam search, top k sampling, and top p sampling to generate text. In our instance, beam search works well because there is no repetitive generation. In top-k sampling, the next words are redistributed among only the k next words after the k most likely words have been filtered. Top-p sampling selects words from the smallest possible set with a cumulative probability greater than p. The configuration of the generation model in our situation was as follows: - temperature was set at 0.6, top k to 64, and top p to 0.8.

We also use the early stopping mechanism to avoid overfitting and found that the model performs best when the epoch number is around 310 using the entire dataset in our example. When an anonymous input sequence is provided to the model, it determines the section of the sentence that is suitable for keyword extraction and utilizes it to generate questions. Due to the fact that the weights were previously set during training, the input sequence is first encoded using the tokenizer and saved in input ids. Then the decoder function generates the questions based on the encoded input sequences.

E. Question Evaluation

The QA evaluator takes a question and answer pair as input and returns a value indicating whether or not the input was a valid question and answer combination. Since there is no solid way of evaluating a question-answer pair because there is no right answer, researchers use a classification model where we corrupt some part of data and assign a respective boolean value to it.

Our model is bert-base-cased with a sequence classification head. This model was trained on the same data as question generation with the context being stripped off. For about half, we have corrupted the answers by either swapping the answer for an unrelated answer or by copying part of the question into the answer. For the wrong question-answer pairs, we assign a value of 0 and 1 for the other pairs. The question-answer pairs generated by the T5 model and the custom model are sent to the evaluator, which ranks all of the QA pairs. The top N pairs with the highest scores are then chosen by the evaluator model and displayed as the output.

IV. RESULT

We trained the question generator T5 model for about 300 epochs and ended up with a validation loss of 2.85. The evaluator model was also trained for approximately 300 epochs, with a final accuracy of 82 percent. Aside from that, we chose 100 questions at random and classified the question-answer pairs based on their syntactic correctness and fluency. We discovered that 76 percent of the pairs' questions were syntactically correct, and our model generated approximately 84 percent fluent questions.

A. Input

```
Bombay, now known as Mumbai, the city of dreams and the city that never sleeps has a
vast history that is turning to obscurity.
Even though many of you don't know the history of Mumbai, but the history of this
dreamy city is something that the Mumbaikars are passionate about.

The name Mumbai is a nickname for Mumbadevi, a local Goddess.
This city with a beautiful and rich heritage is originally built on a group of seven
islands- Colaba, Worli, Mazagaon, Parel, Mahim, Old Woman Island, and Bombay Island.
These islands were embodied in the Maurya Empire which was under the Buddhist
Emperor Ashoka of Magadha.

It was 1543 AD when the islands were seized by the Portuguese from Bahadur Shah of
Gujarat.
The islands were ruled by the Portuguese till 1661. On May 8, 1661, the islands were
yielded as a dowry to Charles II of England who married to Catherine of Portugal.
So it was then when Bombay became a British possession due to a part of the dowry.
```

Fig. 3. Input Text

B. Output

```
2) Q: What is the name of the city that never sleeps?
   A: Bombay, now known as Mumbai, the city of dreams and the city that never sleep
s has a vast history that is turning to obscurity.

3) Q: Who ruled the islands?
   A: The islands were ruled by the Portuguese till 1661.
```

Fig. 4. Generated Questions

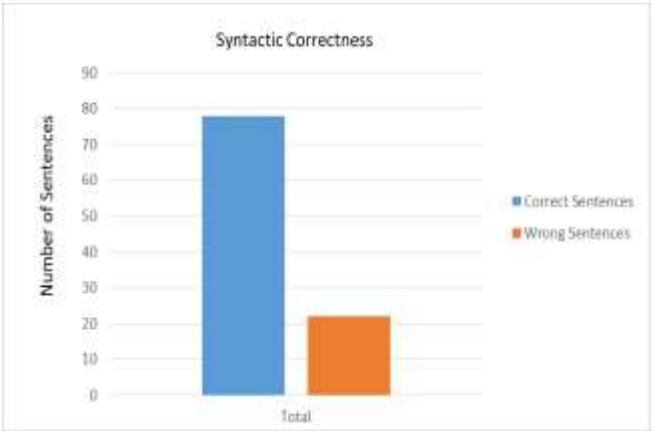

Fig. 5. Syntactic Correctness

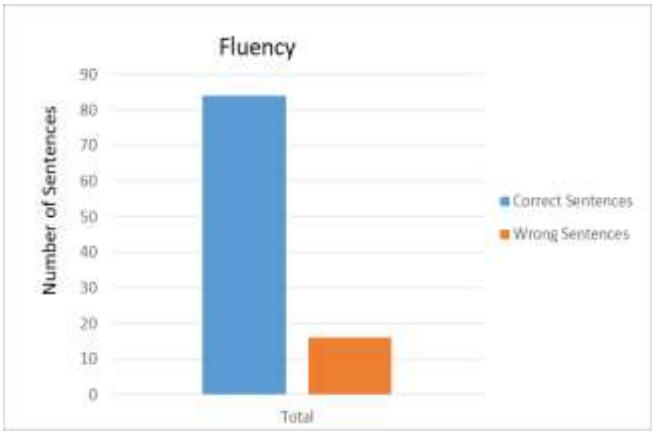

Fig. 6. Fluency Score

V. Conclusion

An amalgamation of state-of-the-art models, such as the T5 transformer, and our proposed tailored model has consistently performed for a wide range of scenarios. When it comes to selecting the highest-quality questions, the evaluator model's performance becomes critical. This quality should not only be in terms of syntactic correctness but also its complexity in context.

It would be critical to have more datasets for the model to train across a range of fields. To work with niche terminology in engineering, medicine, or economics, we'll need a precise custom distractor generation model. A fully automated system like this will reduce regression time, allowing more complex systems like internet search engines and chatbots to emerge.

VI. References

[1] Biswas, Saroj & Bordoloi, Monali & Shreya, Jacob. (2017). A Graph Based Keyword Extraction Model using Collective Node Weight. Expert Systems with Applications. 97. 10.1016/j.eswa.2017.12.025.

[2] Du, X. and Claire Cardie. "Identifying Where to Focus in Reading Comprehension for Neural Question Generation." EMNLP (2017).K. Elissa, "Title of paper if known," unpublished.

[3] Yuan, Xingdi & Wang, Tong & Gulcehre, Caglar & Sordoni, Alessandro & Bachman, Philip & Zhang, Saizheng & Subramanian, Sandeep & Trischler, Adam. (2017). Machine Comprehension by Text-to-Text Neural Question Generation. 15-25. 10.18653/v1/W17-2603.

[4] Sarmento, Rui & Cordeiro, Mário & Brazdil, Pavel & Gama, João. (2018). Incremental TextRank - Automatic Keyword Extraction for Text Streams. 10.5220/0006639703630370.

[5] Soleymanzadeh, Katira. (2017). Domain Specific Automatic Question Generation from Text. 82-88. 10.18653/v1/P17-3014.

[6] Sun, Xingwu & Liu, Jing & Lyu, Yajuan & He, Wei & Ma, Yanjun & Wang, Shi. (2018). Answer-focused and Position-aware Neural Question Generation. 3930-3939. 10.18653/v1/D18-1427.

[7] Qiu, Zhaopeng & Fan, Wei. (2020). Automatic Distractor Generation for Multiple Choice Questions in Standard Tests. 2096-2106. 10.18653/v1/2020.coling-main.189.

[8] Mahdavi, Sedigheh & An, Aijun & Davoudi, Heidar & Delpisheh, Marjan & Gohari, Emad. (2020). Question-Worthy Sentence Selection for Question Generation. 10.1007/978-3-030-47358-7_40.

[9] Roemmele, Melissa & Sidhpura, Deep & DeNeefe, Steve & Tsou, Ling. (2021). AnswerQuest: A System for Generating Question-Answer Items from Multi-Paragraph Documents.

[10] Bartolo, Max & Thrush, Tristan & Jia, Robin & Riedel, Sebastian & Stenetorp, Pontus & Kiela, Douwe. (2021). Improving Question Answering Model Robustness with Synthetic Adversarial Data Generation.

[11] Santhanavijayan, A., Sadhu Ramakrishnan Balasundaram, Srinivas Narayanan, S. Vinod Kumar and V. Vignesh Prasad. "Automatic generation of multiple choice questions for e-assessment." International Journal of Signal and Imaging Systems Engineering 10 (2017): 54.

[12] Bhale, Niranjan L., Sneha Patil, Mansi Pawar, Tanmayi G. Katkade and Nikita D. Jadhav. "Automatic Question Generation Using Wikipedia." Imperial journal of interdisciplinary research 2 (2016): n. pag.

[13] https://rajpurkar.github.io/SQuAD-explorer

[14] https://stanfordnlp.github.io/coqa/

[15] Miller, Derek. (2019). Leveraging BERT for Extractive Text Summarization on Lectures

[16] Trask, Andrew & Michalak, Phil & Liu, John. (2015). sense2vec - A Fast and Accurate Method for Word Sense Disambiguation In Neural Word Embeddings

[17] George A. Miller (1995). WordNet: A Lexical Database for English. Communications of the ACM Vol. 38, No. 11: 39-41.

[18] Raffel, Colin, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li and Peter J. Liu. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer." ArXiv abs/1910.10683 (2020): n. pag.