Weakly Supervised Context-based Interview Question Generation

Samiran Pal, Kaamraan Khan, Avinash Kumar Singh, Subhasish Ghosh, Tapas Nayak, Girish Palshikar, Indrajit Bhattacharya
TCS Research, India
samiran.pal@tcs.com, kaamraankhan1@gmail.com,
{singh.avinash9, g.subhasish, nayak.tapas, gk.palshikar, b.indrajit}@tcs.com

Abstract

We explore the task of automated generation of technical interview questions from a given textbook. Such questions are different from those for reading comprehension studied in question generation literature. We curate a context-based interview questions data set for Machine Learning and Deep Learning from two popular textbooks. We first explore the possibility of using a large generative language model (GPT-3) for this task in a zero shot setting. We then evaluate the performance of smaller generative models such as BART fine-tuned on weakly supervised data obtained using GPT-3 and hand-crafted templates. We deploy an automatic question importance assignment technique to figure out suitability of a question in a technical interview. It improves the evaluation results in many dimensions. We dissect the performance of these models for this task and also scrutinize the suitability of questions generated by them for use in technical interviews.

1 Introduction

Asking good questions is crucial for assessing candidates in technical interviews. But this requires human experts with technical knowledge and experience. Therefore, the capability to automatically generate technical questions to assess knowledge and understanding for a specific subject can significantly reduce expert effort in conducting interviews and in scaling up the interview process. In this paper, we focus on automated generation of interview questions from textbook contexts.

There has a been a lot of interest in recent years on question generation (QG) (Dhole and Manning, 2020; Bang et al., 2019; Back et al., 2021), and many benchmark data sets exist (Rajpurkar et al., 2016). Dhole and Manning (2020); Mazidi and Nielsen (2014); Heilman and Smith (2010) focus on rule-based question generation, wherein Dhole and Manning (2020) transform declarative sentences into question-answer pairs using syntactic rules, universal dependencies, shallow semantic parsing and lexical resources. These generate precise questions but often fail due to their heavy reliance on manually crafted feature sets. Recent papers (Xiao et al., 2020; Zhao et al., 2018; Serban et al., 2016) use deep neural networks for QG. Serban et al. (2016) focus on generating factoid questions from knowledge bases such as Freebase. These use answers as clues to generate questions. Back et al. (2021); Cui et al. (2021); Huang et al. (2021) propose answer-agnostic QG. Back et al. (2021) predict answer-like candidates for the given passage and then generate questions from these. Huang et al. (2021); Tsai et al. (2021) use transformer-based generative models such as BART (Lewis et al., 2020) and T5 (Raffel et al., 2020). In summary, all existing models and benchmark datasets address factoid question generation for reading comprehension (RC).

Generating technical interview questions from a context is harder than generating RC questions. Such questions focus on technical concepts and their relationships and test for knowledge and understanding of these. The answers are long-form (2-5 sentences) and must be contained in the context ('What is regularization?' is not answerable from the context 'This issue can be addressed using L2 regularization'). Questions must be semantically complete ('What is the form of the optimal solution?' is an incomplete question) and of appropriate specificity ('What is machine learning' is too generic for assessing expertise in ML). Finally, questions should have a diverse mix of intent and task complexity (compared to just remembering). As a result, RC-question generation models are not appropriate for interview question generation, and RC-question generation data sets are not useful for evaluating interview question generation models.

To address this gap, we first create a dataset of textbook contexts and corresponding technical interview questions from two popular textbooks on Machine Learning (ML) and Deep Learning (DL). We use this for evaluation of interview question generation algorithms. Generation of large volumes of gold-standard training data for this task is difficult and expensive. So, we first evaluate pre-trained large language models (LLM) such as GPT-3 for this task in a zero-shot setting. We then explore fine-tuning a relatively smaller LM (BART) on weakly supervised training data. Wang et al. (2021) have recently explored GPT-3 for creating training data for many NLP tasks including factoid RC question generation. We explore the use of GPT-3 and template-based algorithms for silver-standard interview question generation from textbook contexts. We improve the question generation quality by developing a post-processing unit which filter out questions automatically by checking its suitability in a technical interview. We evaluate the models before and after filtering and see that it improves the performance in many evaluation dimensions.

Using detailed analysis of interview questions generated using these models, we highlight the challenges of this task, and also the key aspects that future models will need to address.

2 Technical Interview Question Dataset

There is no public dataset on technical questions from textbook contexts for evaluating our task. Available question datasets for reading comprehension such as SQuAD (Rajpurkar et al., 2016) are not ideal. So we create a gold-standard dataset by manually generating questions on Machine learning (ML) and Deep learning (DL) from two publicly available books, Bishop (2006) for ML and Goodfellow et al. (2016) for DL.

We selected 3 chapters each from the ML and DL books. We distributed these chapters to 10 annotators (internal to our organisation) with subject knowledge as well as interview experience. We consider each section in a chapter as one context, and the annotators generate all possible long-form questions from each such context. The annotators are reminded of interview question templates (Sec.3.2) but are not restricted to these. However, they are instructed to ensure answerability from context. Each chapter was first annotated independently by two annotators. Then, annotation differences, i.e., questions generated by one but not the other, are resolved via discussion. A question is added in the dataset only if both annotators are convinced after discussion, thus making the inter-annotator agreement 1 for the final dataset. We obtained 161 questions from 3 chapters of the DL book, and 187 questions from the 3 chapters of the ML book. We discuss quality aspects of the dataset in Sec.4. We will make this dataset public.

3 Question Generation Approaches

Here we describe the different question generation and post-processing technique we use in the paper.

3.1 Zero-shot Generation with GPT-3

GPT-3 (Brown et al., 2020) is a transformer-decoder based large language model (LLM) and has shown excellent performance on many NLP tasks in zero-shot settings (Wang et al., 2021; Yoo et al., 2021). We explore GPT-3 for zero-shot technical interview question generation[1]. We use its interview question generation preset, where we give a context as prompt to GPT-3 and it generates a set of interview questions from the context. At a time, we provide one context paragraph, followed by a new line and an instruction or prompt ('Give a list of questions from above passage'). We experiment with different variants of prompt that include the number of questions we want GPT to generate such as 'Generate 10 questions...'. While such prompts work, we observe that the initial set of questions does not differ if prompts are changed. But the questions in the later part of the generated set changes for different prompts. This suggests some internal ranking of the questions generated by GPT-3. So, we decide not to include the number of questions in the prompt and left this for GPT-3 to optimize. We set the temperature parameter to 0 to eliminate randomness in the generated questions, and the other parameters as default.

We ran this GPT-3 question generation process on the same chapters that were used for human question generation (Sec.2). We do a comparative analysis in Sec.4.

3.2 Generation with Fine-tuned BART

While GPT-3 is very powerful, it comes with a cost in real-world applications. As a free open-source alternative, that can be fine-tuned for our specific task, we explore BART (Lewis et al., 2020), another pre-trained language model that has been used for Question Generation (Huang et al., 2021;

[1]We use free GPT-3 APIs.

Tsai et al., 2021). Since our task is question generation, we use the BART model post-trained on SQuAD. Since SQuAD contains factoid RC questions, this need further fine-tuning for our task. For this, we create a silver-standard dataset. We take the remaining chapters of the two books (11 for ML and 17 for DL), which were not used for creating the gold-standard dataset. We use two approaches for generating silver-standard questions from these - a template-based algorithm and GPT-3.

We use an unsupervised template-based algorithm for QG given the context (Fabbri et al., 2020; Puzikov and Gurevych, 2018; Yu and Jiang, 2021). Such approaches typically have high precision while generating a smaller number of questions. We use the following templates: WHAT IS X?, WHAT ARE ADVANTAGES / DISADVANTAGES / USES OF X?, WHAT ARE THE DIFFERENCES BETWEEN X AND Y?, WHAT IS THE RELATION BETWEEN X AND Y?. We use precise regular expressions and dictionaries for each template to check applicability of a template to a sentence, and use book indices to extract concepts for X and Y. For each context, we take the union of questions generated using each template for each sentence in the context. We get 626 questions in all using this approach. More details are included in the Appendix.

We also use GPT-3 to generate silver-standard questions from the same contexts as described in Sec.3 . This gives us 3,013 questions in total.

We experiment with BART fine-tuned on template and GPT-3 questions separately, as well as together. We combine template and GPT-3 questions generated from the same context in two different ways. For a context C, let Q_t and Q_g be the sets of questions generated by the template algorithm and GPT-3. In the 'Concat' mode, we consider (C, Q_t) and (C, Q_g) as two distinct training instances (rows in the training data). In the 'Join' mode, we create a single training instance $(C, Q_t \cup Q_g)$.

3.3 Importance based Question Filtering

While analysing the generated questions we saw that many of them are trivial or obsolete. For example, 'What is a scalar?' is too basic of a question to be asked in a technical interview. Similarly 'What are the advantages of recirculation?' is also not a pressing question for a technical interview because 'recirculation' is not a very well known or popular concept in machine learning. So it does not reveal much about the expertise of a candidate by asking this question.

We assign importance scores to all the questions generated from the book and filter out questions with importance score below a threshold. We use the hierarchical Index and Table of Content (TOC) of the book for assigning the importance scores. First, we automatically create a concept list, L_c present in the book from the Index and TOC. Using this concept list, we annotate the concepts present in each question. We assign importance score for all the concepts in the question and then assign importance to the question.

We observed that concepts appear more upfront in the book are more fundamental and very often later concepts are dependent on them. So there is a strict partial order among concepts present in the book determined by prerequisites. All books are written in a way that prerequisites of a concept appear before the concept because one has to know the prerequisites to understand the current concept. With this observation, we make an assumption that concepts that appear upfront in the book or in the chapters are more fundamental, hence more important, than the concepts which appear later and dependent on them. With this assumption, we calculate two scores, namely TOC_score and Index_score for each concept to find its importance. TOC_score gives the importance score for a given concept in reference to the TOC. Each concept can appear in multiple chapters, in multiple sections within a chapter and in multiple subsections within a section. For each such occurrences, we have the associated id from the TOC. For example, 'supervised learning' can appear in multiple places in the book and suppose one occurrence has the id 4.7.2. From this id, we can infer that the above concept appears in chapter 4 (chap_id), section 7 (section_id) and subsection 2 (subsection_id). Using this information, we assign a score for a particular occurrence of the concept, c as follows.

$$\begin{aligned} TOC\_occurrence\_score(c) = \\ (chap\_cnt - chap\_id) * 100 \quad + \\ (max\_sections - section\_id) * 10 \quad + \\ (max\_subsections - subsection\_id) \quad (1) \end{aligned}$$

Here chap_cnt, max_sections, max_subsections are total number of chapters, maximum number of sections, subsections in a chapter respectively. As it is apparent, we give highest priority to chapters, then sections in the chapter, and finally the subsections in the section. For each occurrence of a concept we calculate the score and the final TOC_score of a concept is sum of scores from all TOC_occurrence_score and normalised by the max score of all concepts.

Likewise Index_score gives a score to a concept in reference to the Index of the book. The Index is basically a forest where each concept appear once. Here we assume that a concept is more important if has more sub-concept related to it. More specifically, an important concept will have a bigger sub-tree under it considering the concept as root. To calculate the score, we attach a weight $w_i = d_{max} - i$ to each depth i and also count the number of different concepts in each depth i of the sub-tree, say con_cnt_i. d_{max} is the maximum depth of any tree in the Index. Then the Index_score of a concept, c is calculated as follows.

$$\begin{aligned} &Index\_score(c) = \\ &\quad \Sigma_i(con\_cnt_i * 10^{w_i})/max\_index\_score \end{aligned} \tag{2}$$

Here max_index_score is the maximum of index scores of all concepts. So using the above formula we can have an index score for all the concepts present in the index. The final importance score of a concept is the average of the TOC score and the Index score.

Now we calculate the question importance from concept importance as follows: from the annotated concepts present in a question, we get the score for each concepts. We get the sum of concept importance as question importance and normalise the importance score by dividing each of the question score by the maximum score of all questions. Multiplying the question score by 10 makes the range of the question score as 0 to 10.

We analysed the data and found that the most of the useless questions are tagged with importance 0. Questions with importance greater than 0 is useful to variable degrees. So we only filter out all the questions with 0 importance.

Table 2 shows that filtering improves the scores in all dimensions except one: recall. Filtering will never be able to improve recall, it can at-most be equal to the manual one. Here it is decreasing because some of the good questions are being filtered out by our filtering module. But it is tightly bounded in our case. At the worst case, recall decreases by 9 points.

4 Experiments and Analysis

We present experimental evaluation and in-depth quality analysis of questions generated for test contexts by zero-shot and fine-tuned models using gold-standard technical interview questions as a reference. Detailed hyper-parameter settings are included in the Appendix. The results of our analysis before applying the importance-based question filtering are summarized in Table 1. In Table 2, we include the same analysis after doing the importance-based question filtering. We describe our findings below based on Table 1 i.e. before doing the question filtering.

4.1 Analysis of Precision

We consider a question to be a valid interview question from the context if it is (a) long-form, (b) complete, (c) of the right specificity, and (d) answerable from the context. A question is incomplete if it has unresolved co-reference or it misses some context for the question to be meaningful. We manually analyzed all the questions generated by all algorithms for the 3 test chapters of the DL book for these criteria. Performance for these individual dimensions as well as overall precision is shown in Tab.1.

Answerability from Context: For zero-shot GPT-3 questions, 35% do not have answer in context. The following are some example questions with relevant parts of the contexts: (Q:*What is the contrastive divergence training?*, C:"an autoencoder gradient provides an approximation to contrastive divergence training of RBMs"), (Q:*How can the singular value decomposition be used to invert a matrix?*, C:"the most useful feature of the SVD is that we can use it to partially generalize matrix inversion to nonsquare matrices, as we will see in the next section."). These illustrate the difficulty of ensuring answerability. BART performs similarly, in all modes but one. When fine-tuned in the 'Concat' mode, BART scores 73% and GPT-3 scores 65% (in table 1). We conjecture that this improvement in answerability over GPT-3 generated questions is due to two reasons. First, the template-based question generation conditions ensure context relevance, but GPT-3 questions are not. In the 'Concat' mode, we combine the training data from GPT-3 and template algorithms in such a way that one context have two training instances. We believe that this gives more weight to such questions which appear in both questions set and have answer in context. Second, Wang et al.

Model	Tr. Q. Src.	#Questions	% In-Context	% Complete	Prec.	Rec.	% Complex
Manual	NA	161	100	100	1.00	0.61	22
GPT	NA	**236**	65	85	0.53	**0.47**	23
BART	T	75	56	**96**	0.55	0.16	0
BART	GPT	129	63	80	0.55	0.27	34
BART	Concat(T,GPT)	95	**73**	87	**0.64**	0.23	24
BART	Join(T,GPT)	132	61	80	0.52	0.26	**37**

Table 1: Performance of zero-shot GPT-3 (GPT) and various fine-tuned versions of BART using manually curated interview questions as reference **before filtering**. Long-formness and specificity are not shown as all models score 100%. Recall is evaluated by considering all model-generated valid questions in adddition to human questions as reference. Tr. Q. Src.=Training questions source. T=template-based questions, GPT=GPT-3 generated questions.

Model	Tr. Q. Src.	#Questions	% In-Context	% Complete	Prec.	Rec.	% Complex
Manual	NA	161	100	100	1.00	0.61	22
GPT	NA	**155**	66	94	**0.8**	**0.47**	31
BART	T	53	51	96	0.49	0.1	0
BART	GPT	83	70	60	0.64	0.2	**46**
BART	Concat(T,GPT)	60	**75**	**98**	0.72	0.16	30
BART	Join(T,GPT)	79	61	90	0.57	0.17	46

Table 2: Performance of zero-shot GPT-3 (GPT) and various fine-tuned versions of BART using manually curated interview questions as reference **after filtering**.

(2021) proves that under consistency assumption, a model trained using GPT-3 labelled data has lower classification error rate than GPT-3 itself. Here, GPT-3 labelled data acts as a regularizer during training. This result shows that fine-tuning smaller LMs such as BART can achieve better performance in this dimension compared to GPT-3.

Question Completeness: 15% of all GPT-3 questions are incomplete. The following are some examples: *Q:How can the issue of scale be prevented?*, *Q:What are some of the models that this technique can be applied to?*. The percentage is slightly lower for fine-tuned BART in Concat mode (13%), but the nature of mistakes is similar.

Long-form, Specificity, Semantic correctness: We define Long-formness for a question if its answer has more than one sentence. For Specificity, we assume the contribution of the question in Interview setting for assessing the candidate's knowledge, i.e; the question shouldn't be too broad or too specific for a topic. Almost all models score $\sim$100% for long-formness and specificity level of questions. The following is an outlier example of a GPT-3 question that is too specific and simple for an ML interview: *Q:What is the multiplication of a matrix by a scalar?* However, GPT-3 does generate some questions that are semantically invalid: *Q:What is the determinant of a matrix if the determinant is 0?*, *Q:What is Shilov?* (Shilov is the name of a researcher).

Overall Precision: A question is valid if it satisfies all the listed criteria. Precision is the ratio of the number of valid questions and the number of generated questions. The precision of GPT-3 is 53%. Fine-tuned BART in 'Concat' mode has 64% precision — an 11% improvement over GPT-3.

4.2 Analysis of the Recall

Evaluating recall with respect to the human generated questions is problematic because this question set need not be exhaustive. Instead, we manually identify additional valid questions generated by each model from a context beyond those in the human generated set. We take the union of these with the human generated questions as the set of all valid questions for a context. We identify and eliminate equivalent questions when taking the union. We define recall of a model as the ratio of the number of valid generated questions and the total number of valid questions. Recall for the different models is shown in Tab.1.

Among the models, GPT-3 generates the most questions and also has the highest recall (47%). However, in general it fails to generate DEFINE, EXAMPLES OF and WHY questions. The fine-

Model	Tr. Q. Src.	#Questions	Prec.	Rec.	RougeL-Prec.	RougeL-Rec.
Manual	NA	161	–	–	–	–
GPT	NA	**236**	0.44	0.61	0.25	0.33
BART	T	75	0.69	0.34	0.35	0.14
BART	GPT	129	0.56	0.44	0.29	0.25
BART	Concat(T,GPT)	95	0.62	0.39	0.34	0.21
BART	Join(T,GPT)	132	0.55	0.44	0.27	0.25

Table 3: Automatic Evaluation of questions generated from DL book (before importance based filtering) using Mapping and RougeL based precision, recall.

Model	Tr. Q. Src.	#Questions	Prec.	Rec.	RougeL-Prec.	RougeL-Rec.
Manual	NA	187	–	–	–	–
GPT	NA	**297**	0.42	0.60	0.21	0.30
BART	T	127	0.53	0.38	0.27	0.15
BART	GPT	169	0.50	0.46	0.24	0.22
BART	Concat(T,GPT)	143	0.49	0.42	0.26	0.20
BART	Join(T,GPT)	181	0.52	0.50	0.26	0.27

Table 4: Automatic Evaluation of questions generated from ML book (before importance based filtering) using Mapping and RougeL based precision, recall.

tuned BART 'Concat' model has much lower recall (23%). Effectively, GPT-3 has higher F1 (48%) compared to BART Concat (35%).

On the other hand, GPT-3 and other models generate many questions missed by humans, so that human generated questions have a recall of only 61%. One example of GPT-3 questions missed by humans is *What is the pseudoinverse of a diagonal matrix?*. Here, the definition of *pseudoinverse* was hidden inside explanation of mathematical notation in the context. Another is *What is the difference between the singular value decomposition and the eigendecomposition?* from the context "SVD is more generally applicable. Every real matrix has a singular value decomposition, but the same is not true of the eigenvalue decomposition". The human annotator generated a *What is the Advantage of* question, but missed the *What is difference between* question. This shows the potential of model generated questions. However, GPT-3 or fine-tuned BART do not generate any questions according to a new template completely missed by humans.

4.3 Complexity and Diversity of Questions

In an interview, candidates should be asked a variety of both simple and complex questions. We define a question as complex if it contains more than 1 concept from the topic. From Table 1, we see that our BART model in 'Concat' mode, generates similar % of complex questions as GPT-3 zero-shot setting. Beyond validity, diversity and complexity of questions is also important. One simple measure of complexity is the number of concepts covered by the question. 22% of human generated questions have multiple concepts. Among the models, BART Concat as well as GPT-3 have a similar percentage. Interestingly, BART Join generates fewer questions but a higher percentage of multi-concept questions.

The 5 levels of Bloom's Taxonomy (Bloom et al., 1964) provide another definition of cognitive task complexity associated with a question (defined in A.2). Interestingly, the human generated questions are uniformly distributed across these levels. However, both GPT-3 (57%) and BART Concat (58%) have a high proportion of questions from the simplest 'Remember' level, corresponding mostly to WHAT IS questions.

4.4 Automatic Evaluation

Since manual evaluation is extensive and costly, it is not scalable for large set of data. To alleviate such scenario we also provide some automatic measure for the evaluation of generated questions from different approaches. For the given context we already have manually generated questions which can serve as gold standard so our goal is to evaluate the quality of generated questions with respect to manual questions.

Since we have multiple manual and model generated questions for each context, our first aim is to have a one to one alignment for these two sets of questions. For this, lets say there are m manual and n model generated questions for a context, then we create a $m \times n$ matrix where each cell represents the similarity score $sim_{i,j}$ between the i'th manual and j'th generated question. Here the similarity score represents the cosine similarity between the embeddings of the two questions using sentence BERT (Reimers and Gurevych, 2019). We then pass this matrix as input to Hungarian algorithm (Kuhn, 1955) which outputs one to one mapping between manual and generated questions.

Once we have pairwise questions we compute precision and recall by two different methods, namely mapping-based and RougeL-based (Lin, 2004). In the first one, for every context we filter out pairs of questions whose similarity scores are less than a threshold. Then we calculate precision per context by summing the scores of the remaining pairs and dividing it by the total number of generated questions. Likewise recall per context is obtained by dividing it by total number of manual questions. The final precision and recall is the average of all precision and recall per context.

For the second method, we filter out pairs of questions from the outputs of the Hungarian Algorithm like above using a threshold. All the unmapped questions are assumed to be mapped with an empty string. Then we calculate the RougeL precision and recall for every pair of questions. Precision per context is calculated by dividing the sum of RougeL precision by number of generated questions in the context and Recall per context is obtained by dividing sum of the RougeL recall by the number of manual questions. Final precision and recall is the average of all precision and recall per context. Just like manual evaluation, we apply the automatic evaluation on unfiltered questions. We include the automatic evaluation results for questions generated from the DL book in Table 3 whose manual evaluations are included in Table 1. We can see that both the automatic evaluation methods in Table 3 positively correlate with the manual evaluation in Table 1. Automatic mapping based precision and recall have correlation 0.43 and 0.99 respectively with the manual precision and recall. Automatic RougeL based precision and recall have correlation 0.66 and 0.94 respectively.

Seeing the above correlation, we have deployed our automatic evaluation methods on questions generated from 3 chapters of ML book for which we do not have any manual evaluation. We have given the evaluation results for ML questions in Table 4.

5 Limitations

In this work, we are trying to address what is the best way to evaluate the task of question generation for the technical interview. First limitation of this work is that we could not create a large annotated dataset as reference data as we need people with subject expertise for annotation. Due to the same reason, we were forced to train the generative models like BART on silver-standard dataset generated using GPT-3. As GPT-3 is not freely available, the amount of silver-standard data is also limited in volume. The evaluation dimensions we propose for this task involve human effort and obviously this is not scalable.

6 Ethical Impact Statement

Generating interview questions using LLMs may have some ethical concern when these questions are used in actual interviews. In this work, we propose how to evaluate such questions in multiple dimensions. People should be careful before using such questions in actual interviews.

7 Conclusion

In this paper, we explore the problem of technical question generation for interviews from textbook contexts in an weakly supervised approach. We curate a dataset for evaluation for two domains, machine learning and deep learning, using two popular books. We analyzed zero-shot question generation using GPT-3 and fine-tuning BART on silver-standard training data for the same. We also suggested a post processing filtering unit which further improves the quality of generated questions. Our manual analysis brings out the complementary strengths and weaknesses of these approaches for this task. More importantly, our detailed error analysis highlights the challenges of the task and shows a pathway to better models by identifying the major types of errors to focus on. Finally we devised automatic evaluation techniques which positively correlate with our manual evaluation.

References

Seohyun Back, Akhil Kedia, Sai Chetan Chinthakindi, Haejun Lee, and Jaegul Choo. 2021. Learning to generate questions by learning to recover answer-containing sentences. In *FINDINGS*.

Liu Bang, Zhao Mingjun, Niu Di, Lai Kunfeng, He Yancheng, Wei Haojie, and Xu Yu. 2019. Learning to generate questions by learning what not to generate. In *The World Wide Web Conference*.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

Benjamin Samuel Bloom, Committee of College, and University Examiners. 1964. *Taxonomy of educational objectives*, volume 2. Longmans, Green New York.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *NeurIPS*.

Shaobo Cui, Xintong Bao, Xinxing Zu, Yangyang Guo, Zhongzhou Zhao, Ji Zhang, and Haiqing Chen. 2021. Onestop qamaker: Extract question-answer pairs from text in a one-stop approach. In *ArXiv*.

Kaustubh D. Dhole and Christopher D. Manning. 2020. Syn-qg: Syntactic and shallow semantic rules for question generation. In *ACL*.

Alexander Fabbri, Patrick Ng, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2020. Template-based question generation from retrieved sentences for improved unsupervised question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Michael Heilman and Noah A. Smith. 2010. Good question! statistical ranking for question generation. In *ACL*.

Xinting Huang, Jianzhong Qi, Yu Sun, , and Rui Zhang. 2021. Latent reasoning for low-resource question generation. In *ACL*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Harold W Kuhn. 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and com- prehension. In *In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Karen Mazidi and Rodney D. Nielsen. 2014. Linguistic considerations in automatic question generation. In *ACL*.

Yevgeniy Puzikov and Iryna Gurevych. 2018. E2E NLG challenge: Neural models vs. templates. In *Proceedings of the 11th International Conference on Natural Language Generation*.

Colin Raffel, Noam Shazeer, Adam Roberts, Kather ine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. In *Journal of Machine Learning Research, 21(140):1–67*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *ACL*.

Danny C.L. Tsai, Anna Y.Q. Huang, Owen H.T. Lu, and Stephen J.H. Yang. 2021. Automatic question generation for repeated testing to learning outcome. In *ICALT*.

Shuohang Wang, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. 2021. Want to reduce labeling cost? gpt-3 can help. In *EMNLP*.

Dongling Xiao, Han Zhang, Yukun Li, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie-gen: An enhanced multi-flow pre-training and fine-tuning framework for natural language generation. In *IJCAI*.

Kang Min Yoo, Dongju Park, Jaewook Kang, Sang-Woo Lee, and Woomyeong Park. 2021. Gpt3mix: Leveraging large-scale language models for text augmentation. In *EMNLP*.

Xiaojing Yu and Anxiao Jiang. 2021. Expanding, retrieving and infilling: Diversifying cross-domain question generation with flexible templates. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*.

Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. 2018. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *EMNLP*.

A Appendix

A.1 Template based question generation

After going over real interview transcripts, we identify some templates for the questions and write algorithms for each of the template. We use these templates for this task. We include some of these templates and their examples in Table 5.

We run these templates on the sentences of the books to generate the context-based question. We use different algorithms for each template. Here, we describe the algorithm for 'Difference' template. Each template algorithm uses a list of concepts and a list of keywords. Let C_list={c1, c2, c3, ...} be a list of concepts present in the book. We created this list using the index and Table of content of the books. Multiple atomic concepts can combine to form a maximal concept. For ex: "convolution" and "neural network" can combine to "convolution neural network". There can be multiple maximal concept in a sentence which we store in a list denoted by Mx_c_list. Other than concepts present in the sentence we also look for template specific keywords or phrases which either give hints for attributes of a concept or relationship between two concepts. we call these phrases signals. For ex, presence of "*difference between*" in a sentence hints the possibility of distinctions between two concepts being discussed. So we can generate a 'Difference' question from this sentence. We define a function Is_signal_present(sentence,signal_list, X,Y) which will take a sentence of a book, signal lists and concepts X, Y and returns true if signal along with concepts X and Y are present in the sentence to generate a question out of it. We have shown our pseudo code in algorithm1. Here, X and Y refers to some concepts in the topic for which questions are generated. We include some examples of template generated questions from context in Table 6.

A.2 Bloom's Taxonomy

Bloom's taxonomy defined 6 categories of knowledge in terms of skills/abilities in an increasing order of cognitive load. We loosely assign different types/templates of questions with one of these Bloom's categories: (i) Remember ('What is X?') (ii) Understand ('How does SUB V X?'), (iii) apply ('What are some applications of X'), (iv) analyze ('Explain X'), (v) evaluate ('What are the differences between X and Y'), (vi) Create (We do not have any questions in this category). Here, X and Y refer to the concepts of the topic. As an example 'cross-entropy loss' is a concept in machine learning.

Algorithm 1 *Difference* template algorithm

```
for sentence in context do
    $Mx\_c\_list$ = find_maximal_concept (sentence, C_list)
    for $X$ in $Mx\_c\_list$ do
        for $Y$ in $Mx\_c\_list$ do
            $flag$ = Is_signal_present(sentence, signal_list, $X$,$Y$)
            if $flag$ then
                Q_gen: What is the difference between $X$ and $Y$?
            end if
        end for
    end for
end for
```

A.3 Question Importance Algorithm

We have described the method to calculate the question importance method in 3.3. Here we provide the pseudo-code for the same in algorithm 2.

A.4 Parameter Settings

We experiment with BART-base (Lewis et al., 2020) model post-trained on SQuAD (Rajpurkar et al., 2016). All experiments are done on 64GB CPU with 20 cores. We use a batch size of 16 and train the model for 10 epochs. The average time to train for 10 epochs is around 2 hours. We optimize the model parameters using Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.0001. Some contexts in our dataset is longer than what BART encoder can accommodate (1,024 word piece tokens). During training, we use the first 1,024 token of the context and discard the remaining part of the context. During inference, we split the longer context into multiple chunks of 1,024 tokens and run inference on each one of them. We do an union of the generated questions from all the splits.

Template	**Example**	**#Questions**
What is X?	What is EM algorithm ?	394
What are some uses of X?	What are some uses of maximum likelihood?	118
What are the advantages of X?	What are the advantages of the validation set?	66
What are the disadvantages of X?	What are disadvantages of the metropolis algorithm?	32
What are the differences between X and Y?	What are the differences between bagging and boosting?	4
What is the relation between X and Y?	What is the relation between autoencoders and latent variables?	12

Table 5: Examples of the template generated questions using some of the templates. Here, X and Y refers to some concepts in the topic for which questions are generated.

Context	**Questions**
An auto encoder is a neural network that is trained to attempt to copy its input to its output.Internally, it has a hidden layer h that describes a code used to represent the input. ... Recently, theoretical connections between auto encoders and latent variable models have brought auto encoders to the forefront of generative modeling.	What is an auto encoder? What is the relation between auto encoders and latent variable?
One advantage of directed graphical models is that a simple and efficient procedure called ancestral sampling can produce a sample from the joint distribution represented by the model. ... Ancestral sampling is generally very fast (assuming sampling from each conditional is easy) and convenient. One drawback to ancestral sampling is that it only applies to directed graphical models. Another drawback is that it does not support every conditional sampling operation. ...	What is ancestral sampling? What are advantages of ancestral sampling? What are disadvantages of ancestral sampling?
Unlike the deep belief network (DBN), it is an entirely undirected model. Unlike the RBM, the DBM has several layers of latent variables (RBMs have just one). ... A DBM is an energy-based model, meaning that the joint probability distribution over the model variables is parametrized by an energy function E . ... In comparison to fully connected Boltzmann machines (with every unit connected to every other unit), the DBM offers some advantages that are similar to those offered by the RBM. ...	What are some use of the dbm? What are differences between the rbm and the dbm? What is a dbm?

Table 6: Examples of generated questions from the context using corresponding template matching algorithm. We match the template algorithm at the sentence-level in the context.

Algorithm 2 Question Importance algorithm

```
$concept\_importance$ = {}
for concept in L_concept do
    $toc\_score$ = 0
    $max\_score$ = $-1$
    $occurance\_list$= all_occurances(concept)
    for occurrence in $occurance\_list$ do
        $toc\_score$ += occurance_score(occurance) {from equation 1}
    end for
    $toc\_score$ = normalise($toc\_score$)
    score = average($toc\_score$, $index\_score$) {from equation 2}
    $concept\_importance[concept]$= score
end for
$question\_score.fromkeys(L\_concept, 0)$
for question in question_list do
    $concept\_list$= get_concepts(question)
    for concept in concept_list do
        $question\_score[question]$+= $concept\_importance[concept]$
    end for
end for
$max\_score$ = find_max($question\_score$)
for question in question_score do
    $question\_score[question]$= $question\_score[question] * 10/max\_score$
end for
return $question\_score$
```