| Title / Year / Authors | Description |
| --- | --- |
| [Question Generation by Transformers](https://arxiv.org/pdf/1909.05017.pdf) / 2019 / Kriangchaivech, Wangperawong | Researchers developed a method for automatically generating questions using transformers. They use a seq2seq method, and claim that transformers are both faster to train and better performing for this task than RNNs. They trained the system on the SQuAD dataset, which they inverted by providing the text of various Wikipedia articles and answers to questions, from which their model would attempt to infer questions. They used spaCy for named entity recognition and WordPiece for tokenization. They graded their questions using word error rate (*WER*) between generated questions and the corresponding questions in SQuAD. This gives rise to the equation , where S is the number of word substitutions, D is the number of word deletions, I is the number of word insertions, and N is the total number of words in the reference sequence.  The researchers found that the vast majority of questions began with ‘What’, meaning that there may not be much complexity in the question types generated using this method. Additionally, they found question pairs with low *WER* but high semantic difference and words with high *WER* but low semantic difference. This means that there is likely another method better suited for grading question quality. |
| [MOOCCube: A Large-scale Data Repository for NLP Applications in MOOCs](https://aclanthology.org/2020.acl-main.285.pdf) / 2020 / Yu et al. | Researchers introduce MOOCCube, which is a repository of more than 700 online courses, 100k concepts, and 8 million student behaviors. The researchers extracted valuable information from each of the courses, like class synopses, teachers, and organizational information. Additionally, the researchers extracted the 10 most representative course concepts and built a taxonomy relating them called a concept graph. |
| [MOOCCubeX: A Large Knowledge-centered Repository for Adaptive Learning in MOOCs (aminer.cn)](https://lfs.aminer.cn/misc/moocdata/publications/mooccubex.pdf) / 2021 / Yu et al. | An extension of the MOOCCube paper listed above, the authors introduce MOOCCubeX. It is a repository with a similar setup to MOOCCube, but with many more courses and with an emphasis on adaptive learning. This emphasis on adaptive learning could be really important for our project. It allows for questions to be better tailored to the skill level of individual students rather than just a question bank of similar difficulty questions. The development of MOOCCubeX is made up of three stages: data processing, fine-grained concept acquisition, and data curation. In the data processing the researchers did things like removing duplicate courses and classification of courses. In step two, fine-grained concept extraction, the researchers developed a method that required little supervision to extract concepts from course material, discover relationships between them, and build a concept graph. Lastly in data curation the researchers conducted more concept annotation and focused on intra-course connections. The most promising portion of these works for us is the methods they utilize for key concept extraction. By identifying the key concepts of a given text, it will be possible to ensure our question generation techniques cover the breadth of the important concepts in a given text. |
| [Towards Generalized Methods for Automatic Question Generation in Educational Domains (cmu.edu)](https://www.cs.cmu.edu/~hn1/papers/ECTEL2022_TowardsGeneralized.pdf) / 2022 / Huy et al. | This is the baseline work for our Focal project from last year. It introduces the methods that we will begin with in our research for this semester. For the purposes of this article, the pipeline introduced stopped at the question generation portion, whereas we will attempt to extension to answer generation and evaluation in this semester. The pipeline introduced for the automatic generation of questions took part in two major processes: concept extraction and question generation. For concept extraction, they too utilized the MOOCCubeX pipeline. The researchers would input the text for a given section into the MOOCCubeX pipeline, and in return would receive a list of the key concepts. A researcher would then manually eliminate the invalid results, such as identified concepts that were generic words and not truly concepts of the given text. After extracting the key concepts for the text, researchers began the question generation step. For this process, they applied Google’s T5, which they had pre-trained on the SQuAD 1.1 dataset.They would also prepend the input text to T5 with either the unit, module, or topic name, which they found increased the number of cogent, pedagogically sound questions. After generating the questions, the researchers used what they call information score to grade the generated questions. This score essentially calculated the number of tokens in a generated response that coincided to a key concept from the text and divided by the number of tokens so as not to bias their score to longer questions. After calculating information score the authors also utilized GPT-3 to predict the pedagogical soundness of a generated question. They fine tuned GPT-3 on the LearningQ dataset. Lastly, the authors had domain experts evaluate whether generated questions were pedagogically sound or not. In all, the researchers found that 56.7 % of the 203 generated questions were deemed pedagogically sound, compared to GPT-3 rating 74.38% sound. Researchers found that typically, GPT-3 tended to overestimate the soundness of questions. Additionally, information score is an area for improvement in future research. In most cases, the information score of expertly evaluated questions was not significantly different between sound and unsound questions, meaning it may not be of great value for evaluating generated questions. . |
| [[2105.11698] Guiding the Growth: Difficulty-Controllable Question Generation through Step-by-Step Rewriting (arxiv.org)](https://arxiv.org/abs/2105.11698) / 2021 / Cheng et al. | In this research, the authors demonstrate a method for varying the complexity of automatically generated questions. They determine that rather than defining complexity of a question in terms of the ability of an automatic question answering model’s ability to correctly answer it, it should instead be defined as the number of inference steps required to answer it. Their proposed framework has the ability to iteratively increase the complexity of generated questions by rewriting them under the guidance of an extracted reasoning chain. They used a dataset built from HotPotQA and compared their success to that of four baselines. Their model consistently performed better than other question generation techniques when manually graded by evaluators for format, conciseness, answerability, and answer matching. They do not propose a method for automatic question evaluation, which would be a useful addition to their research. |
| [Transformer based end-to-end question generation](https://d1wqtxts1xzle7.cloudfront.net/86040635/2005.01107v1-libre.pdf?1652752372=&response-content-disposition=inline%3B+filename%3DTransformer_based_End_to_End_Question_Ge.pdf&Expires=1677175830&Signature=U6KzbrhuuDNdhOMXqqp~sE6b6dqx4pEvnHJzsDKReUo1jKNtbfxP0o3VEJ0G~tX3iQBnd09ES-qhjKGy4VRLPme4spEQvsI1q-pePgp-O4pB03UI1C1e~oKv9CRoU6YimtvUKV3ldDJDDnjBFoky~zQtXukMXRVmVRkcPS2fDjsivnBo23yVgB1oCTRygV-p-JsKQukAzephY0JAm~5~DnfwOgHrLB-gsNo2m9puqn7RBzUcvznM11JTRjf6b~VfDMsMDpiN9LGzq7LQTrqAtrKDwkDBDpA9XRHCdD3xTFFXyd7X0tkJVnWbU5fC5BPwUYqUU6IqeYs2ODOS8XRfVQ__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA) / 2020 / Lopz et al.  Cont. | In this research, the authors attempted to demonstrate that transformer-based finetuning can be used to create a robust question generation system that only uses a single language model without the need for extra mechanisms or features. They used Hugging-Faces 124 million parameter GPT-2, from which they finetuned 6 question generation models using different variations of data combinations. They used a variety of metrics to grade produced questions, including BLEU-1 through BLEU-4 and METEOR. While these methods are sufficient for determining similarity between input text and the generated questions, there is no inherent evaluation for the pedagogical soundness of the questions. For example, some of the generated questions simply restated the input text. So while it had a high valued score on similarity evaluations, it may not have produced much actual educational value. |
| [Automatic Question-Answer Generation from Video Lecture using Neural Machine Translation | IEEE Conference Publication | IEEE Xplore](https://ieeexplore.ieee.org/document/9566139) / 2021 / Sonia et al. | In this paper, the researchers are demonstrating a method for automatic question and answer generation from a video. They extract text and feed it to their model, which is pretrained on the SQuAD dataset much like the other methods discussed for question generation. |
| [Design and Development of a Framework for an Automatic Answer Evaluation System Based on Similarity Measures (degruyter.com)](https://www.degruyter.com/document/doi/10.1515/jisys-2015-0031/html) | In this article the authors introduce a framework for automatically evaluating the correctness of a student’s provided solution. This framework essentially acts by pre-processing the provided answer and comparing its cosine similarity to a reference solution. Their technique involves comparing the connection between the given answer and reference solution based not only on the entire text, but also on similarity in specific parts of speech. These researchers provide a strong method for evaluating student responses in our pipeline, provided the answer generation technique we utilize sufficiently produced enough answers to cover the breadth of correct solutions. |
| [Automatic Generation of Programming Word Problems](https://www.cse.msu.edu/~wangzh65/AI4EDU/papers/18.pdf) | The ability to write and solve programming word problems is an essential skill for computer science students. However, generating these problems requires a significant amount of manual effort. In this paper, the authors present a method for automatically generating simple and realistic programming word problems. The proposed method involves synthesizing a problem in a high-level graph-based notation and then incorporating it into a story. The authors define a graph-based representation for a program and create a set of templates for each node in the graph. They then organize and combine the templates to generate both simple narrative and story-based programming problems. The proposed method was tested with human judges, and the results showed that the generated problems are sufficiently realistic. This work is related to previous research on generating programming exercises, but it offers a different approach that can generate diverse and challenging algorithmic exercises. |
| [Easy-to-Hard: Leveraging Simple Questions for Complex Question Generation](https://arxiv.org/pdf/1912.02367.pdf) | The paper proposes a system for automatically generating complex questions from knowledge graphs (KGs), which aims to translate KG representations into universally understandable natural language questions. The authors first design a base model, CoG2Q, which is an encoder-decoder neural network following the methodology of state-of-the-art simple question generation (QG) work. The authors then propose two extensions, CoGSub2Q and CoGSubm2Q, to improve the performance of CQG by leveraging existing simple questions. CoGSub2Q encodes and copies from a sub-question, while CoGSubm2Q scores and aggregates multiple pseudo sub-questions to generate the final complex question. Experimental results show that the extension models significantly outperform the base CoG2Q, demonstrating the importance of instance-level connections between simple and corresponding complex questions. |