Focal Design Document

Students:
Razik Grewal, Peter Meyers, Mitali Potnis

Mentors:
Annette Han, John Stamper

Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Version 1.4

1 Introduction

As education continually trends toward online learning, giving students frequent assessments is crucial to their development.[1] These assessments often come at a high price to educators however, as they are forced to create a large bank of questions to provide realistic learning opportunities for students of varied skill levels. [2] In addition to the high cost to produce these assessments, educators likewise must spend a significant amount of time providing students with quality feedback. This feedback helps to ensure the assessments students are completing enrich their learning experience, but is time-consuming to produce. Focal intends to reduce the burden of these assessments on educators, thereby saving both their efforts and time. By automating the assessment creation and evaluation pipeline, educators' time will be freed to focus on their various other responsibilities.

Given our intent to make a product for use by educators and students, our system design is intended to make Focal a tool that provides assessments in an end-to-end manner. Educators should be able to upload their course content and receive questions that are both cogent and pedagogically sound. Likewise, students should receive these questions and immediate useful feedback on their responses. The pipeline should generate questions, answers, and response evaluations without external input from domain experts. While Focal is eventually intended to consist of these functionalities in conjunction with a user interface for educators and students, the user interface is outside of the scope of this year-long research iteration. Instead, we will focus on providing the end-to-end pipeline in a repository for future development.

2 Design Considerations

When establishing our solution for Focal, we considered the following key design areas: assumptions, constraints, system environment, and design methodology.

1. **Assumptions**: One of the primary assumptions that is necessary for Focal to be a viable solution is that the various courses hosted on OLI have similar content layouts. While there is expected to be some variation in layout between courses, if courses vary significantly in their organization, it may limit Focal's ability to be domain-agnostic. Additionally, we are assuming that various educators have the same views of what assessments are useful for their students. Because Focal creates questions automatically and without input from domain experts (in this case educators), there may be some questions that specific educators

do not find useful. We are operating under the assumption that these differences are not so great as to significantly impact usability for educators. Lastly, we are assuming that a single Focal pipeline can provide utility in a variety of educational domains. Even with differences in material, we assume that a single pipeline can provide cogent, pedagogically sound questions.

2. **Constraints**: A primary constraint of Focal that was mentioned in the requirements document is that it must be both fast and highly accurate. In regards to widespread adoption and utility, students and teachers expect a solution that provides assessment with minimal delay. Additionally, the questions that are generated must be consistently high quality. If educators find question quality to be sub-par, they are much more likely to stop using the Focal pipeline. Additionally, students answers must be accurately graded to ensure their motivation to use the system is not hampered.
3. **System Environment**: When complete and functional, Focal will be integrated in the Open Learning Initiative (OLI) user interface. OLI hosts various online courses at a number of institutions across the United States. The content used for Focal's development comes from OLI, so it naturally will be integrated with its user interface. This integration will occur after the completion of this year-long capstone however. At the completion of this research project, Focal will exist in a Github repository for future researchers.
4. **Design Methodology**: As stated earlier, Focal is designed to be an end-to-end assessment generation tool. This means that educators can simply provide the course content and Focal will do the rest. Focal will take in the course content and generate questions that it will provide to students. Student will receive those questions through the OLI user interface and respond accordingly. Lastly, Focal will evaluate these student responses and provide the educator and student with feedback.

3 System Architecture/Design Overview

As discussed in the design methodology, Focal's primary workflow is to take in course content, produce questions, and evaluate student responses. These primary steps can be further broken down into the following steps, which were introduced in Focal's requirements document:

1. An educator uploads course material to the user interface.
2. Material is processed by Focal's concept extraction model and the proposed key concepts are returned to the educator for review.
3. If the educator approves the list of key concepts, they are sent to Focal's question generation model.
4. If the educator does not approve of the list of key concepts, they are given the opportunity to add or subtract any they disagree with. After these revisions, the list of concepts is sent to Focal's question generation model.
5. Focal's question generation model takes in the list of key concepts and course material and returns to the educator a list of relevant questions.
6. If the educator approves of the list of questions, it is sent to Focal's question answering model as well as to the user interface for student interaction.
7. If the educator does not approve of the list of questions, they are given the opportunity to add or subtract any they disagree with. After these revisions, the list of concepts is sent to Focal's question answering model as well as to the user interface for student interaction.
8. Focal's question answering model creates a list of answers for each question in the approved questions list. These answers are sent to Focal's answer evaluation model.
9. Student's receive questions in the user interface and respond accordingly. Their responses are sent to Focal's answer evaluation model.

10. Focal's answer evaluation model compares the generated answers to student responses and evaluates them for correctness. These evaluations are then sent to the educators and students for review.

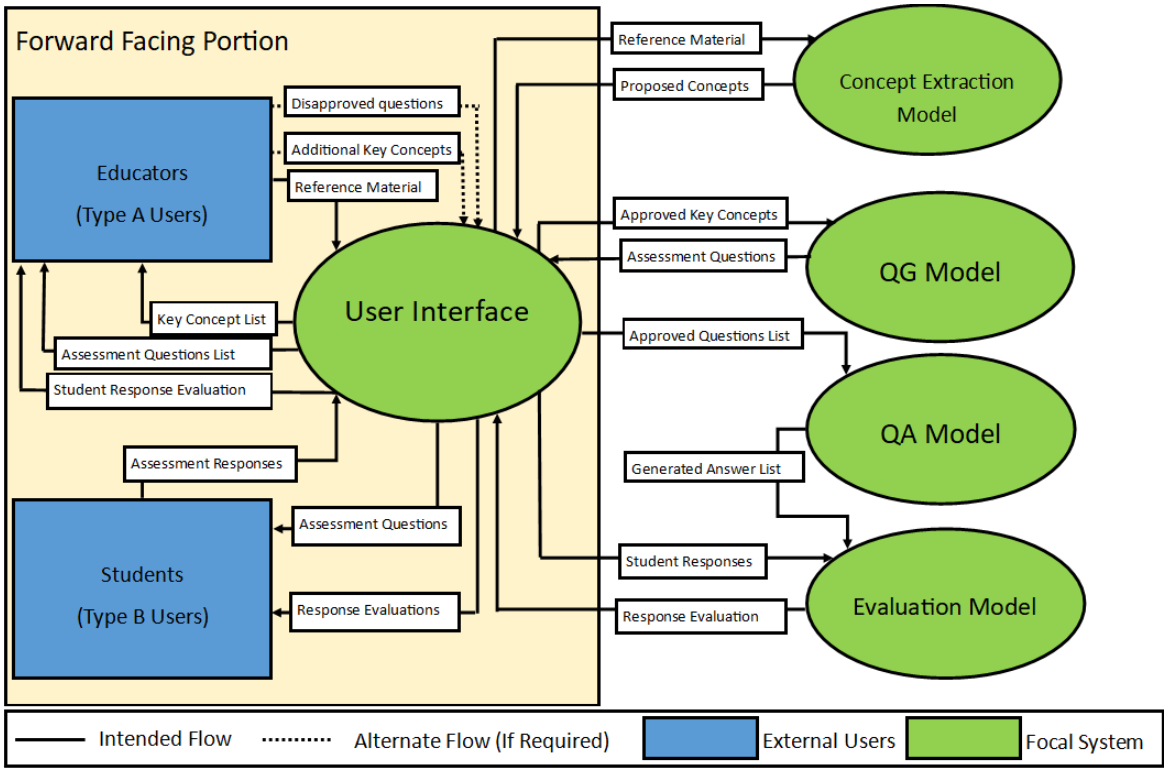

Fig. 1 Focal architecture diagram, with an operational user interface, which would be housed in OLI.

4 Data Design

For the purposes of the Focal pipeline, our data will be textual data from a graduate-level data science course and an undergraduate-level chemistry course. The data will exist in XML format, which our pipeline will parse into usable text. After Focal parses the input text, it will be sent to our question generation and question answering models for further use in the pipeline.

Within the course content text, there are multiple hierarchical levels: Units, Modules and Topics. Previous iterations of research on Focal have demonstrated the value of pre-pending course content with the name of its hierarchical categories when providing the content to their question generation model. [2] As this proved valuable we intend to pre-pend the unit, module, and topic titles as well. This pre-pending is done for each separate hierarchical level, for example the unit title is concatenated with the text content, which provides one section of input text to the question generation model. The following is a graphical depiction of this pre-pending process:

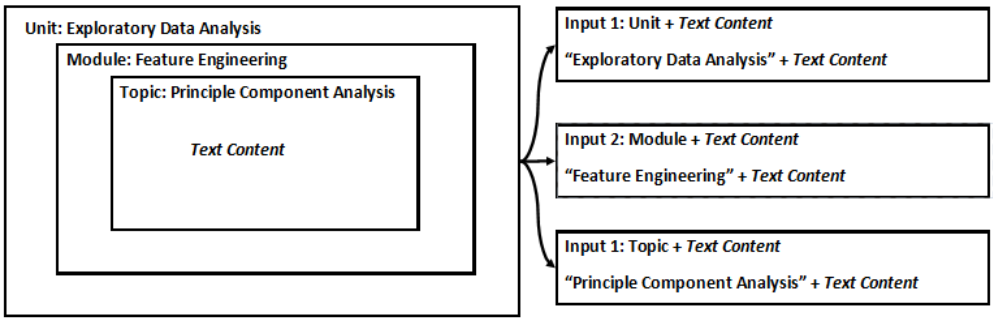

Fig. 2 Graphical depiction of data pre-pending process demonstrated to be valuable in generating high quality questions. [2]

5 Implementation Overview

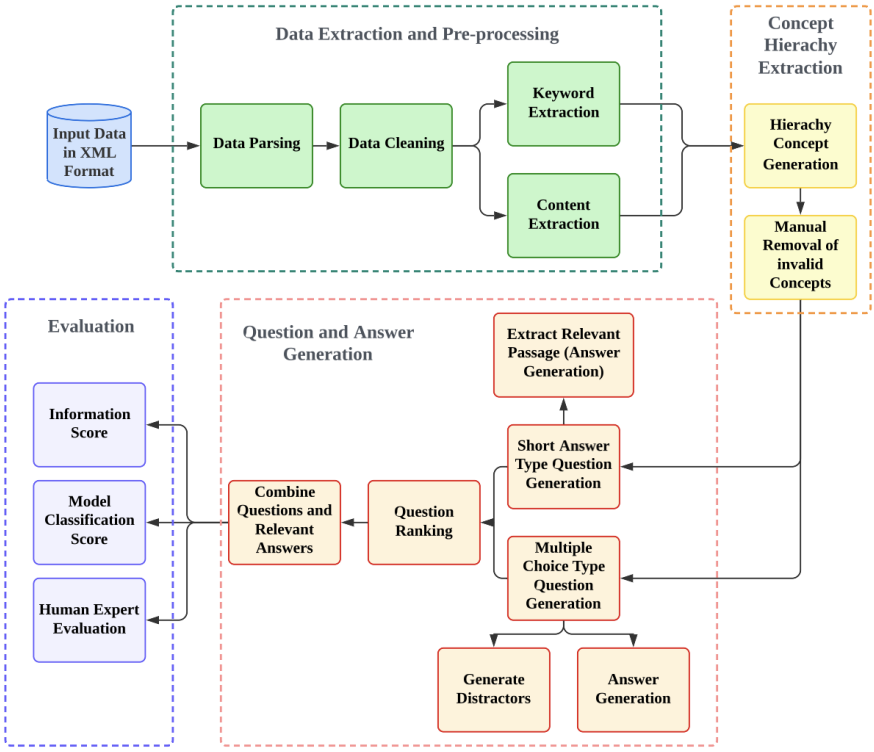

Fig. 3 System Workflow.

We will also be using data for fine-tuning our models involved in question-generation and evaluation of our generated questions. In the case of fine-tuning our question generation model, we are using the SQuAD 1.1 dataset [3]. The SQuAD 1.1 reading comprehension dataset involves around 100,000 reading comprehension questions based on Wikipedia articles. For finetuning our evaluator model we utilize the LearningQ dataset [4]. The LearningQ dataset contains over 230 thousand document-question pairs and 7000 instructor-designed questions on a variety of educational topics. No pre-processing is carried out for these publicly released datasets.

The entire question-answer generation workflow is illustrated in figure 3. On the user side, once the user answers the questions, the workflow for evaluating user's answers can be seen in figure 4.

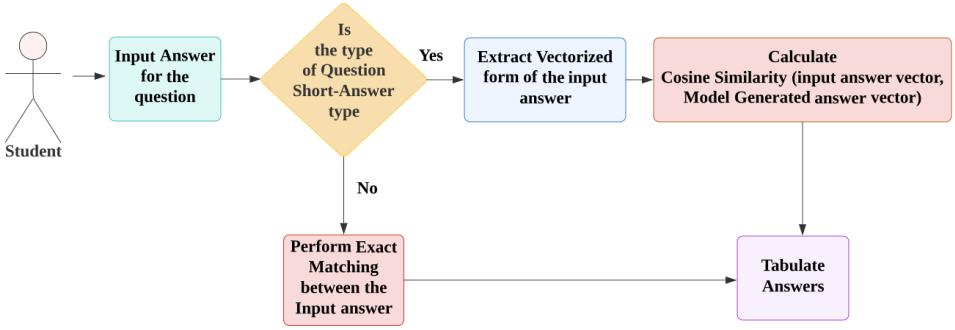

Fig. 4 User Answer Evaluation Workflow.

6 Design Models

The overall system workflow seen in fig. 3, comprises of 4 sections: Data Extraction and Pre-processing, Concept Hierarchy Extraction, Question and Answer generation, and Evaluation of the generated Questions and Answers. The first section, Data Extraction and Pre-processing involves scraping the input XML data using the BeautifulSoup library. We further clean the data which involves steps like removing the punctuation marks and remove stopwords.Extraction of headings, keywords, and text content for each of the topics in the modules of the units is carried out. The resulting text is then passed on to two phases, Concept Hierarchy Extraction and Question and Answer Generation. The Concept Hierarchy Extraction involves implementation of MOOCCubeX pipeline for carrying out weakly-supervised fine-grained extraction of concepts on the input corpus without any expert assistance.[5] Invalid concepts like prepositions, indexing numbers, and generic verbs are then removed after detailed reviewing of the concepts generated. The next phase involves the generation of two types of questions: Multiple-choice questions and short-answer type questions. These questions are generated on the basis of pre-pending topic or module or unit headings to the text for each topic using a fine-tuned transformer-based encoder-decoder T5 model. [6] In case of multiple-choice questions, the distractors for the multiple-choice options are also generated. The relevant answers are then extracted using a custom rule-based-approach involving dependency parse tree We had previously tried extracting answers from the text content using a custom rule-based approach with a dependency parse tree. After ranking the questions and extracting top 3 questions for each topic, the questions and generated their answers are then evaluated to score their pedagogical quality using Information score, model (GPT-3 classification), and human expert evaluation.
On the User side, once the user inputs the answers for the questions, the system evaluates the correctness of the input answer as per the 4. For each of the question, on the basis of the question type, either cosine similarity is calculated or exact matching is carried out. In the case of short-answer type questions, we first generate vector of the input text either using bag-of-words method or tf-idf. We then calculate a cosine similarity score between this vector and the vector of the model generated answer to measure the correctness. In case the question is numerical type, Yes-No type or Multiple-choice Question, we perform exact matching in the input user and the model generated answer to measure correctness of the user's answer.

7 Test Design

Describe the procedure to test the results' models, metrics, data scheme, etc. Then, present the results in a result table. In order to test the pedagogical soundness (if the question generated is related to the course content or is it vague or unrelated to the domain information) of the questions and answers generated, three methods are employed: Information Score, Model classification, and Human Expert Evaluation. In the case of assessing the quality of the answers given by the students, we use cosine similarity and exact keyword matching.

Information Score: The Information score is a custom metric that measures the relevancy of each questions in the context of the identified key concepts (C) using Concept Hierarchy Generation. For any generated question (q), the tokens set T(q) is extracted. The information score is then calculates on the basis of the coinciding of the number of tokens in the question with that of the concept. Higher the information score, the better is the quality of the generated questions in terms of testing student's knowledge regarding key concepts. Equation below denotes the calculation of the Information score.[2]

$$\text{Information Score(q)} = \frac{1}{|T(q)|} \sum_{t \in T(q)} 1(t \in C)$$

Human Expert Evaluation: Subject matter experts with high teaching experience in the domain involved are also used to assess the system's quality in generating pedagogically sound questions. In case of multiple human experts during evaluation, the Inter-Rater Reliability (IRR) can be measured between the experts involved to measure the agreement between the experts.

Model Classification: A Classification model like that of GPT-3 can be used for classifying (rating) if the question is pedagogically sound, or is vague, unclear or unrelated to the domain. The GPT-3 model will be fine-tuned and further can be evaluated to generated binary class labels of ratings of pedagogically sound or not sound.

Cosine Similarity and Exact Matching: For assessing the correctness of the answers given by the students, we will be performing exact matching between the student's answer and the original accurate multiple-choice option. In the case of short answer type questions, we will match the cosine similarity between the vectors of the answer given by the user and the relevant text paragraph chosen by our system beforehand. The equation below denotes the cosine similarity between two vectors A (original relevant short answer generated by the user) and S (answer given by the student). The vector forms of the two answers can be generated using either bag-of-words or TF-IDF weighting.

$$\text{Cosine Similarity(A,S)} = \frac{D}{|A| * |S|}$$

where D is the Dot product between the two vectors A and S. On the basis of a pre-defined threshold, we can observe if the cosine similarity is higher than the threshold, to grade the input answer as correct. In the case of Yes-No or Numerical type short-answer questions, we can perform exact matching between the student's answer and the original answer generated by our model.
The Table 1 and 3 below illustrates our testing methods in a tabular format. We will also be comparing the evaluation classification done by our model (GPT-3) and the human expert in the form of a classification matrix seen in Table 2.

Table 1: Evaluation of generated questions across different header levels and soundness ratings

Generated Question	**Header Level (Topic/ Module/Unit**	**Information score**	**GPT-3 Classification (Sound/Unsound)**	**Expert Rating (Sound/Unsound)**
-	-	-	-	-

Table 2: Confusion Matrix for comparing GPT-3 and expert evaluations.

	Expert: Not Sound	**Expert: Sound**
GPT3: Not Sound	-	-
GPT3: Sound	-	-

Table 3: Metric Value for Questions.

Type of Question	**User answer**	**Evaluation Strategy(Exact Matching/ Cosine Similarity)**	**Metric Value**
-	-	-	-

8 Deployment Model

In its final form, Focal will be deployed through the Open Learning Initiative (OLI). OLI is an online learning platform that hosts course content for a variety of online classes. In this form, it will be seamlessly integrated with the course content such that students will have the ability to answer questions that are placed at the end of each section of textual content. This functionality exists in OLI in its current form, with the exception that the questions are entered by educators, not automatically generated. With Focal, this process will be automated, giving each student valuable assessments without input from educators. When students respond to assessments, their answers will automatically be evaluated and they will immediately receive valuable feedback.

While Focal is intended to eventually be integrated directly in the OLI platform, that is beyond the scope of the Focal project for this year long iteration. In the case of our research, our deployment will reside in a Github repository with all the relevant documentation to enable future improvements. This repository will consist of the pipeline itself, as well as the course content for the courses used in our research and all relevant documentation.

9 Risks/Challenges

As with any data science project, there are several risks that we must plan for in advance to ensure they do not cause excess problems later on during project execution.

1. **Diverse Course Content**: One major process risk that may jeopardize the schedule is the course content of the additional course we are adding to the Focal pipeline. We are currently working under the assumption that content used in various Open Learning Initiative classes has similar structure, meaning that adding additional courses would not require large modifications to the Focal pipeline. If courses are structure in much different manners, however, it could cause a significant delay in the development of the pipeline. In order to mitigate this risk, we are attempting to get this new course data as soon as possible to begin working further on determining the differences and similarities between courses.
2. **Improper Questions/Evaluations**: As with any new technology, adoption typically relies on the product consistently performing as expected. This is no different for Focal, and it is the primary product risk. If teachers find that the questions students are receiving from Focal are nonsensical or not pedagogically sound, they are much less likely to adopt it for use in the classroom. Additionally, if students feel their answers are being evaluated unfairly or inaccurately, they are likely to become frustrated with the product and it may lose educational value. In order to mitigate these risks, we plan on employing various tests in the Focal pipeline that can grade question quality. Additionally, we plan to test multiple models for answer evaluation to ensure student responses are being fairly graded.
3. **Question Variability**: One of the key project risks for Focal is the difficulty of creating questions of varied types and difficulties. While the previous Focal pipeline created a significant number of cogent questions, they were primarily "what" questions rather than "why" or "how" questions. Creating more difficult and varied questions is non-trivial and has the potential to cause setbacks to the project schedule. To mitigate this risk, we are beginning testing on question generation techniques early on in the project. This will allow us to adjust course if needed while time still remains.

10 Tools and Dependencies

Version of Python: Based on the libaries and their versions involved we will be using Python version 3.7.

Libraries and packages:

- Numpy[version=1.19.5]: Used for numerical operations and manipulation of arrays and matrices containing textual data.
- TensorFlow[version=1.15]: Used for building, training, and evaluating deep learning models such as the neural network models used in the Focal pipeline for question generation and answer evaluation.
- Pandas[version=1.5.3]: Used for data analysis and manipulation of tabular data such as the metadata associated with the course content.
- BeautifulSoup[version=0.12.2]: Used for parsing the XML input data containing the course content.
- gensim[version=4.3.1]: Used for keyword extraction to generate key concepts for use in the question generation process.
- sklearn[version=1.2.2]: Used for feature extraction and vectorization of text data for use in the question generation process.
- matplotlib [version=3.4] and seaborn[version=]: Used for data visualization and analysis of the input data.
- pipelines[version=2.7]: Used for deploying and running the question answering model.
- transformers[version=2.1.0]: Used for fine-tuning and evaluating the transformer-based encoder-decoder model used in the question generation process.
- nltk[version=3.8.1]: Used for natural language processing tasks such as tokenization and part-of-speech tagging.

Datasets involved: In our case, we have two types of datasets involved. The first is the course content text of the graduate-level data science course and an undergraduate-level chemistry course in the XML format. This data is further parsed to get headings and the underlying text. The second type of datasets are those that we are using for fine-tuning our models involved for question-generation and evaluation of our generated questions. In the case of fine-tuning our question generation model, we are using the SQuAD 1.1 dataset [3]. This dataset is a reading comprehension dataset that comprises around 100,000 reading comprehension questions based on Wikipedia articles. For finetuning our evaluator model we utilize the LearningQ dataset [4] that contains over 230 thousand document-question pairs and 7000 instructor-designed questions on a variety of educational topics.

Pre-existing code repositories: The previous iteration's code repository is stored on a private github repository which has been shared with the team, but due to privacy issues, cannot be linked or disclosed in this document.

Hardware: The hardware requirements for the Focal question generation system will depend on the scale of the system and the size of the datasets being processed. For small-scale systems, processing relatively small datasets, a single CPU with at least 4-8GB of RAM and 100GB of storage is sufficient. An AWS EC2 t2.medium instance is an example of a virtual machine that can be used in this case. For medium-scale systems processing larger datasets, a more powerful machine with multiple CPUs and 16-32GB of RAM may be required. An AWS EC2 m5.large instance is an example of a virtual machine that can be used in this case. For large-scale systems processing very large datasets, a distributed system with multiple nodes may be necessary. Each node should have multiple CPUs, at least 64GB of RAM, and hundreds of GB of storage. An AWS EC2 r5.8xlarge instance is an example of a virtual machine that can be used in this case.

The specific hardware requirements for the Focal question generation system will depend on the size and complexity of the datasets being processed, as well as the specific needs of the system being built. Cloud-based services like AWS provide a flexible and scalable infrastructure that can be used to quickly provision and scale up/down resources as needed.

11 Terminology, Definitions, Acronyms, and Abbreviations

- **GPT-3**: Generative Pre-Trained Transformer 3, this product is a large language model that has proven valuable at tasks vital to this project, such as question answering and text generation.
- **MOOCCubeX**: A large dataset of educational materials, in addition to a pipeline that is adept at automatically extracting key educational concepts from text.[5]
- **NLP**: Natural language processing, this is the branch of machine learning that focuses on the processing and generation of natural language.
- **QA**: Question Answering, this is the process of using NLP models to automatically generate answers to given questions.
- **QG**: Question Generation, this is the process of using NLP models to automatically create questions based on text supplied to the model.
- **SQuAD**: Stanford Question Answering Dataset, this is a large dataset made of questions on Wikipedia articles and their corresponding answers.[3]
- **Bag-of-words**: Text representation technique that transforms given text into a bag or set of its words without considering the order of the words and grammar to analyze the frequency of each word in a corpus.
- **Tf-idf**:Term Frequency-Inverse Document Frequency is a statistical measure for evaluating importance of a word in a document on the basis of its frequency in the document and its rarity in the involved corpus for identifying the most relevant words in any given document or corpus.
- **T5**: Test-to-Text Transfer Transformer created by Google, considered a state-of-the-art model in many NLP tasks.[6]
- **LearningQ Dataset**: A large dataset covering 230 thousand document-question pairs and 7000 instructor-designed questions on a variety of educational topics.[4]

12 References

[1] Yong Zhao, Jing Lei, Bo Yan Chun Lai, and Hueyshan Sophia Tan. What makes the difference? a practical analysis of research on the effectiveness of distance education. *Teachers College Record*, 107(8):1836–1884, 2005.

[2] Huy A. Nguyen, Shravya Bhat, Steven Moore, Norman Bier, and John Stamper. Towards generalized methods for automatic question generation in educational domains. In *Educating for a New Future: Making Sense of Technology-Enhanced Learning Adoption*, pages 272–284, Cham, 2022.

[3] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Conference on Empirical Methods in Natural Language Processing*, 2016.

[4] Guanliang Chen, Jie Yang, Claudia Hauff, and Geert-Jan Houben. Learningq: A large-scale dataset for educational question generation. *Proceedings of the International AAAI Conference on Web and Social Media*, 12, Jun. 2018.

[5] Jifan Yu, Yuquan Wang, Qingyang Zhong, Gan Luo, Yiming Mao, Kai Sun, Wenzheng Feng, Wei-Hao Xu, Shulin Cao, Kaisheng Zeng, Zijun Yao, Lei Hou, Yankai Lin, Peng Li, Jie Zhou, Bingsheng Xu, Juan-Zi Li, Jie Tang, and Maosong Sun. Mooccubex: A large knowledge-centered repository for adaptive learning in moocs. *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021.

[6] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mt5: A massively multilingual pre-trained text-to-text transformer. In *North American Chapter of the Association for Computational Linguistics*, 2020.

13 Reflection

In general, the process of creating this document was relatively seamless. From conversations with our research advisor, he encouraged us to follow a similar workflow to past Focal baselines, as it would be a good starting point. From there, we were able to rationalize the best workflow that would seemingly meet the intended goals of Focal. We selected the design of Focal based on the fact that it provides the intended functionalities with minimal redundancies and in a straightforward way. As we go through the process of testing throughout the year-long capstone, we have discussed the possibility of changing the specific models chosen if another model proves more adept at the task assigned, but the overall structure is expected to remain unchanged. If we could do this design document again, it would be useful to have had a little more time for doing some initial testing with these alternate models, but givent he time constraints, we chose the models we believe most liekely to succeed.

14 Changes to Previous Deliverables

The only change to previous deliverables to report is that we have updated the final deployment condition for Focal to be integrated in OLI. We mentioned in previous deliverables that Focal would have its own user interface, but we cleared that up further in this document by stating that the user interface would reside in OLI.

15 Change Log

Table 1. Change Log and Versioning

Version	Date	Changelog
V1.0	2023.03.10	Added Introduction, Design Considerations
V1.1	2023.03.11	Added System Architecture/Design Overview, Data Design
V1.2	2023.03.12	Added Implementation Overview, Design Models, Risks/ Challenges, Terminology
V1.3	2023.03.13	Modified Implementation Overview, Added Test Design, Deployment Model, Tools and Dependencies
V1.4	2023.03.15	Updated Data Design and Tools and Dependencies, References