A FRAMEWORK FOR AUTOMATICALLY GENERATING QUESTIONS FOR TOPICS IN DISCRETE MATHEMATICS

by

Salar Houshvand

December, 2020

Director of Thesis: Dr.Venkat Gudivada, PhD

Major Department: Computer Science

Automated question generation is critical for realizing personalized learning. Also, learning research shows that answering questions is a more effective method than rereading the textbook multiple times. However, creating different types of questions is intellectually challenging and time-intensive. Therefore, it emphasizes a necessity for an automated way to generate questions and evaluate them. In this research after analyzing the existing approaches to automated question generation, we conclude that most of the current systems use natural language process techniques to extract questions from the text, therefore, other topics such as mathematics are lacking an automated question generation system that could help learners to assess their knowledge.

In this research we present a novel framework that automatically generates unlimited numbers of questions for different topics in discrete mathematics. We created multiple algorithms for various questions in four main topics using Python. Our final product is presented as an application programming interface (API) using Flask library, which makes it easy to gain access and use this system in any future developments. Finally, we discuss the potential extensions that can be added to our framework as future contributions. The repository for this framework is freely available at `https://github.com/SalarHoushvand/discrete-math-restfulAPI`.

A FRAMEWORK FOR AUTOMATICALLY GENERATING QUESTIONS FOR TOPICS IN DISCRETE MATHEMATICS

A Thesis

Presented to The Faculty of the Department of Computer Science

East Carolina University

In Partial Fulfillment of the Requirements for the Degree

Master of Science in Software Engineering

by

Salar Houshvand

December, 2020

A FRAMEWORK FOR AUTOMATICALLY GENERATING QUESTIONS FOR TOPICS IN DISCRETE MATHEMATICS

by

Salar Houshvand

APPROVED BY:

DIRECTOR OF THESIS:	Dr.Venkat Gudivada, PhD
COMMITTEE MEMBER:	Nasseh Tabrizi, PhD
COMMITTEE MEMBER:	Nic Herndon, PhD
CHAIR OF THE DEPARTMENT OF COMPUTER SCIENCE:	Venkat Gudivada, PhD
DEAN OF THE GRADUATE SCHOOL:	Paul J. Gemperline, PhD

Table of Contents

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Motivation

One of the best ways to evaluate learning is answering questions. As stated in [38] questions are effective in providing information retrieval from learners' memory, reveal misconceptions, specify the important parts of the material and repeat the highlights. However, relevant questions for every learning article are not always available and designing those questions is a tedious activity for instructors. The automated question generators (QG) are one of the recent approaches to solving this issue. One of the first question generation systems was introduced by Wolfe in 1976 [41] and since then developers proposed diverse methods to construct QG modules. Numerous studies have been done in the QG field, however, the majority of them are discussing question generation systems for the natural languages. In this approach we focused on a system that produces questions for discrete mathematics.

Discrete mathematics is one of the critical and also complex courses for computer science students. It takes more practice to completely understand the concepts and implement them in solving problems. Therefore, instructors need to provide more tools for their students to assess their knowledge, however, creating and providing proper components is a time-consuming activity. The necessity for an automated system that can perform the question asking and evaluating stage of the learning process is unavoidable. The recent advancements in web services are encouraging the developers to create products that utilizes them which are called web applications. Therefore, having a system that can produce questions and be easily accessible for future developers to use in their systems inspired us to implement our proposed method as an application programming interface or API. The proposed system will be developed in Python using Flask as the main framework. Multiple functions are going to be created for processing mathematical operations and generating questions. After the creation of questions, the system returns them in JSON format. This study focuses on answering the following research questions on question generation systems and their construction.

- **RQ1:** What are the existing approaches to question generation systems?
- **RQ2:** What areas are covered by question generation systems?
- **RQ3:** How can we implement a question generation as an application programming interface?
- **RQ4:** How can we develop multiple algorithms to generate questions for discrete mathematics?

1.2 Contribution

This study is conducted to assist at the conscious competence stage of learning by automatically generating practice questions. As stated earlier, in this paper by utilizing programming patterns and mathematical logics, different algorithms have been generated to reduce the involvement of human in creating practice questions. We believe this system would not only be a tremendous help for students to practice their knowledge, but also could help instructors to save time in making those questions.

The approach used in this study began with the identification of four major topics in discrete mathematics and possible subtopics of them. Possible questions extracted from each subtopic and initial patterns were defined for them. Finally, by merging mathematical logic and programming functions, algorithms are created for different questions.

The output of each algorithm is presented in JSON format and collected in a REST-API. The questions are organized using individual topic ids, furthermore, the system assigns a unique id to each question being generated in order to help their reuse in future projects.

1.3 Thesis Outline

This thesis is organized as follows: In chapter 2 we are reviewing existing question generations for natural languages and mathematics as well as different types of questions. We also describe Bloom's taxonomy and its application on questions. We then analyze different parts of an API and how it works in chapter 3. In chapter 4 we bring all our learning together to present our proof of concept. We then provide the requirements and implementation process of auto question generation for discrete mathematics using Flask. Also we go through each topic that we covered in the framework and our API end-points. We conduct our evaluation and represent the results in chapter 5. Finally, we provide suggestions for future work and our conclusion in chapter 6.

Chapter 2

Related Work

Question generation systems cover various topics in science. The prominent area for research in QG systems is natural languages (NLP). Because of the importance of natural languages in question generation systems, we discuss the different approaches and existing frameworks in this section. Then, we review currently available mathematical question generation systems, however, first we introduce Bloom's taxonomy in order to define importance of each question in different stages of learning.

2.1 Relationship Between Bloom's Taxonomy and Question Types

First introduced by Benjamin Bloom, the taxonomy defined as a hierarchical model for the cognitive domain in 1956 [7], after decades, a group of cognitive psychologists, lead by Lorin Anderson made some changes to the taxonomy and made the revised Bloom taxonomy [24]. The purpose of this model is to provide a framework to classify the classroom lesson objectives as well as develop critical thinking and higher order cognitive abilities in students. In this study we are analyzing the revised Bloom taxonomy [24]. In a study by Scott [37] he analyzed cognitive domain in Bloom taxonomy in computer science testing. He concluded that testing of different levels in Bloom's categories will test for student mastery of a subject.

We want to analyze the relation between each attribute of Bloom's taxonomy

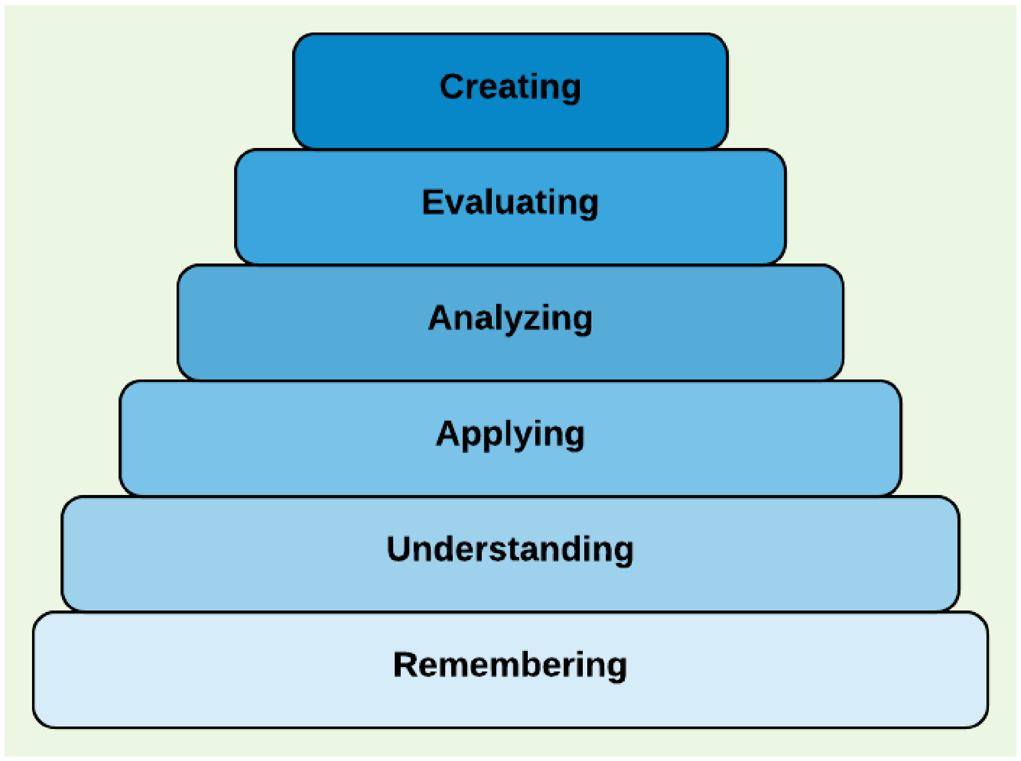


Figure 2.1: Different Stages of Blooms Taxonomy. The taxonomy defined as a hierarchical model that classifies different levels of learning process.

to the question generation. As seen in Figure 2.1 the first step is “Remembering” which consists of define, duplicate, memorize, repeat and state. In this phase we can occupy multiple choice questions to make students identification of what they have studied stronger. As we move forward students need to be more involved in the third phase we have “Applying” which can be described as execute, implement, demonstrate and operate, therefore, we need to design questions that learners can implement the knowledge, descriptive questions would be the best option for this phase. The fourth stage of Bloom’s taxonomy is defined as “Analyzing” and can expand to differentiate, organize, relate, compare, distinguish, experiment and test.

The question needs to offer multiple options to compare, one of the best options would be drag and drop questions or the questions that a student needs to connect choices from right to related choices from left. Fifth phase is “Evaluating” and can be described as appraise, argue, judge, support, value and weight. In this stage student should evaluate and make a decision about what is right, Therefore, the best kind of question for this phase would be judgmental questions which are usually paragraphs of texts or a problem that has already been solved and student needs to read and check, then, decides whether the solution is appropriate or not. Finally, the last stage is “Creating”. At this stage, student is responsible for designing the question, therefore, we don’t have any specific question type for this phase.

2.2 Question Generations Systems using Natural Language Processing Techniques

The main communication tool for human beings since ancient times is language. Interacting with a natural language in computer environment makes the experience more exciting for the user. The growth of computer applications directly affects how peoples’ lives improve as well. One of the major fields that has been altered by computers and artificial intelligence is education.

Finding innovative and effective methods to enhance the learning process has always been a challenging effort for developers. Question answering is one of the best ways to evaluate the knowledge [38] . Therefore, developing systems that can automatically extract questions from learning material is an important task for improving educational field using modern technologies. In 2.2.1 and 2.2.2 we are analyzing several existing approaches and processes in automated question generation in natural languages field.

2.2.1 Process of Generating Questions

Process of making questions is the most important part of each system. Choosing right libraries, algorithms and methods can directly affect questions quality and quantity. As seen in Figure 2.2 question generation process for natural languages consists of multiple step. We are going to discuss each of them.

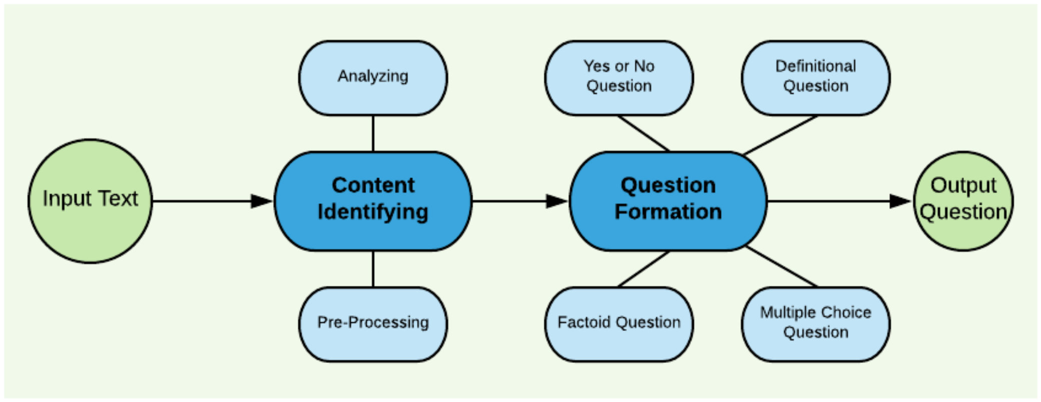


Figure 2.2: QG Systems Architecture for Generating Questions from Text.

Content Identification

Considering semantic complexities, natural languages are hard to process by computers even when they occupy a standard syntactic formation. Moreover, most of the question generation systems function based on predefined linguistic rules. Therefore, in order to get accurate outputs from the module, input sentences should be modified before implementing them into the system. This procedure is discussed in two categories of pre-processing and analyzing the input text.

Pre-Processing

The first step in QG systems is processing and categorizing sentences. One of the most adopted methods is to transform complex sentences into elementary ones, it is done by first parsing the sentences. Parsing is the most used technique to split the given text before the process. The most common tool for parsing is NLP stanford parser [30] is a statistical text parser that defines the grammatical structure of the given sentence and extracts the subject and the objects of the verb. This program has been used in many QG modules such as [22] and [6].
After parsing, sentences are simplified based on specific elements. In a study by [6], sentences have been simplified based on syntactic information to avoid the disambiguation in human language. In a similar way, [22] and [35] have simplified the sentences at the first phase. Heilman and Smith [20] have done this by extracting the textual entailments from the given text.

Since generated elementary sentences should be prepared for analyzing. Part of speech (POS) tagger and Named Entity (NE) tagger are used to identify each word by tagging them their role in the sentence. Following is an example for POS tagger for the given sentence “the weather was so cold.” by using Stanford tagger [30]. As seen bellow DT stands for determiner, NN stands for noun, VBD stands for verb past tense, RB stands for adverb, and JJ stands for adjective.

$$the\_DTweather\_NNwas\_VBDso\_RBcold\_JJ.\_.$$

Most of the time, texts may deliver diverse data and simplifying them may cause loss of important semantic information [31]. In order to prevent that and generate appropriate questions, the module needs to extract more sentences from the given text. In a novel approach by Hilman et al [20] they focused on extracting concise simple sentences from complex inputs based on semantic entailment and presuppositions. They implement semantic entailment by removing adjunct modifiers and discourse connectives such as: non-restrictive appositives, non-restrictive relative clauses, parenthetical, participial modifiers of noun phrases, verb phrase modifiers offset by commas, modifiers that precede the subject and also by splitting conjunctions of clauses and verb phrases. By adopting this method they could more sentences, more accurately from the main input.This is one of the few approaches that focused on the single stage of pre-processing rather than the whole process of question generation.

Analyzing Sentences

Because of the complicated structure of natural language, analyzing the sentences is the hardest step to be considered. There are diverse methods that have been used in this stage of the process.

In English, elements of the sentences can be simplified to nouns, verbs and prepositions and based on those, objects and subjects are defined as well to determine the factors of the algorithm. Ali et al [6] used these elements to classify sentences, after parsing the input, subject, object, verb and preposition extracted by using part of speech tagger and named entity tagger and used those information to classify sentences. Based on the classifications they extract the construction of the sentence and choose the right question type.

Some of the QG systems are searching for a target in the sentence, that could be a relative pronoun as in [23], or keywords as in [22], or discourse cues like [5]. These systems are described as follows:

Relative pronouns and adverbs could be one of the possible question sources. In a study, Khullar et al [23] defined three rules based on nine relative pronouns and adverbs to generate questions. This system choose the wh-word, which is a function word that defines the question such as which, when and etc, from the given input sentence, therefore, the output questions have more accuracy. This method can be used for designing both factoid and definitional questions. The major issue with this method is limitation of sentences that contain relative pronouns and adverbs. Based on the chosen corpus for this study, only 20 percent of the sentences were qualified for this system. Here are examples presented in the study using each of the rules for the given sentence: *"I am giving fur balls to John who likes cats"*

$$Rule1 : Who/Whom\ am\ I\ giving\ fur\ balls\ to?$$

$$Rule2 : Who\ likes\ cats?$$

$$Rule3 : Who\ is\ John?$$

In a different approach by Agrwal et al [5], the role of discourse cues in defining questions has been studied. They chose discourse connectives like *Since, When, Because, Although, for example, for instance* and *as a result*, then analyzed the role of these connectives in defining relations in sentences. They classified relations as *Casual, Temporal, Conditional, Contrast, Concession, Result* and *Instantiation.* Two arguments used to acquire the connectives, these arguments were the main source of question content, however, selecting a right argument was tricky, they defined rules for targeting arguments. Using this method, they were be able to make question types of *when, why, give example* and *yes-no.* These connectives contained average of an 47.01 percent of their chosen corpus. Given the example sentence *"Single wicket has rarely been played since limited overs cricket began"* the module has defined the relation as *Temporal* and then generated the following question:

Since when has single wicket rarely been played?

Some sentences contain appositive phrases, these are noun phrases that give explanations about the noun phrase they precede. At the pre-process step these phrases are removed to simplify the input text, however, they can be a question source as well. In [22] these phrases used to generate questions. They extract these sentences from the input sentence, then, after performing question word identification, they generate the question. As an example from their research in the sentence *"Mexico City, the biggest city in the world, has many interesting archaeological sites"* the module has generated the following question:

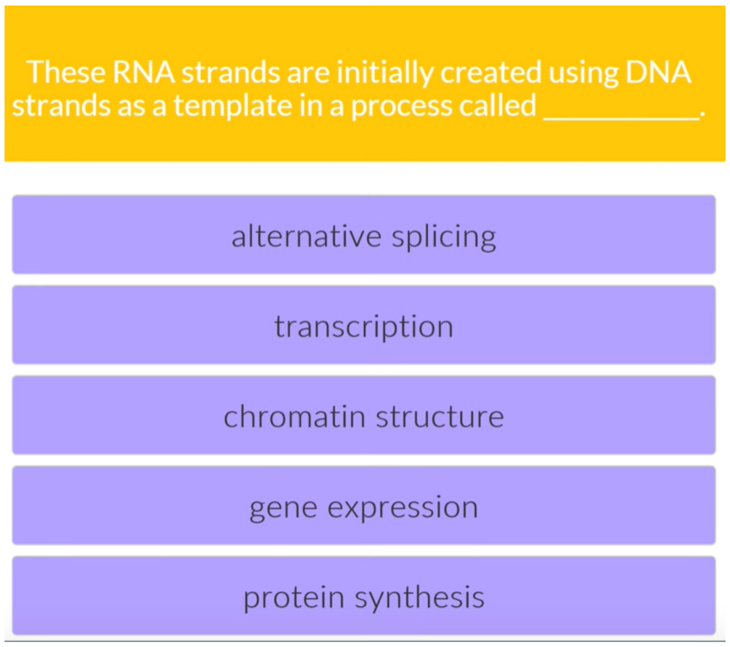
Which/Where is the biggest city in the world?

2.2.2 Question Formation

Key-phrase Questions

Key-phrase or gap-fill questions have short answers that are usually extracted from the keywords of the sentence. In order to make these kinds of questions, a descriptive sentence is being selected from the input text. Next step is choosing the keyword, term frequency is the major way to find it [13]. Summarization features such as number of nouns and pronouns, length of a sentence and number of common tokens are considered in these types of questions [4].

One of the main examples of gap-fill question generators would be RevUP [26]. This is an automatic gap-fill question generation which generates questions in three steps: sentence selection, gap selection and multiple choice distractor selection. The data that is collected by human contribution is used to train discriminative classifiers in order to extract the gap-phrases. Figure 2.3 is an example question from RevUP system.



These RNA strands are initially created using DNA strands as a template in a process called __________.

alternative splicing

transcription

chromatin structure

gene expression

protein synthesis

Figure 2.3: An Example Question from RevUP System.

Definitional Questions

Definitional questions have long descriptive answers and are being used to analyze understanding of concepts that need to be explained. Types of questions that can be used in these concepts are mostly wh-questions. Definitional questions or deep learning questions can be occupied in the "Apply" section of the Bloom's taxonomy. One of the best approaches to this kind of QG is automatic question generation system for discussion questions by Adamson et al [3]. Following are some example questions generated by their system:

- Why does psychological manipulation unite the animals against a supposed enemy ?
- Whose idealism leads to his downfall?
- What does the increasing frequency of the rituals bespeak? Discuss in detail.
- Who gathers the animals of the Manor Farm for a meeting in the big barn?

Another example for definitional question is a study from liu et al [29], which presents a domain independent automatic question generation tool that generates questions which can be used as a form of support for students to revise their essay. This module is based on semantic and syntactic information acquired from citations. They defined five rules with patters based on reporting verbs and sentiment words. As stated in [29] the first rules pattern is defined as "The predicate verb matches reporting verb for expressing author's opinion purpose.". Templates for questions that are based on rule one are as follows:

- Who is [Author Name]?
- What does [Author Name] [predicate verb Lemma]?
- In the [Author Name]s study, do you agree that [Author Name] [Predicate]? Have you evaluated [Author Name]s opinion?
- How did you present [Author]'opinion as evidence to confirm the thesis in your essay?
- Is [other Author Name] against [Author Name]'s opinion? Since you say [Other Author Name]s opinion is against [Author Name], can you find the contradictory evidence provided by [Other Author Name]?

Based on the template, the questions generated for the sentence "*Graddol on the other hand points to the social and economic inequality that the dominance of English could lead to.*" are listed below:

- Who is Graddol?
- What does Graddol point to in his study?(Sourcing)
- Why would Graddol point to the social and economic inequality that the dominance of English could lead to? (What evidence does Graddol provide to prove that?) (Sourcing)
- How did you present Graddol opinion as evidence to confirm the thesis in your essay?(Integration)
- Is Crystal against Graddol's opinion?

- Since you say Crystal's opinion is against Graddol, can you find the contradictive evidence provided by Crystal? (Integration)

True or False Questions (TFQ)

This type of question asks for the correctness of the given clause and has a binary answer. There are pros and cons for TFQ. One of the best advantages is that these questions are easy to generate and also can be easily evaluated [10]. However, true or false questions have limitations on their assessments for students, such as, the probability of guessing the right choice is high (50%) [10].
There are two ways to generate these types of questions, first one is by simply giving a sentence and asking to evaluate the truth of the sentence which can be done by extracting sentences from the text and using comparison to evaluate whether it is true or false, second method is by using subject-auxiliary to make yes or no type questions [22]. After the pre-process and analyzing the sentence, the subject of the verb is defined. Then by placing auxiliary verb in front of the subject, yes or no questions are made.
The important part in making yes or no questions is choosing the correct form of auxiliary verb grammatically. The tools can be used for this purpose are Tregex [28], Tsurgeon and WordNet [32] [22].

In a study by [15] they have designed an automatic TFQ generator based on the keywords. The module first finds a true informative sentence from the text, therefore, it looks for the keyword and after changing the keyword with its antonym, makes the false statement. Finally, all the sentences are stored in a database and make the questions.

Multiple Choice Questions MCQ)

Making multiple choice questions has different challenges, two of which are selecting the right keyword to be questioned and using proper choices and distractors in order to make the question accurate and also fairly hard.

There are different ways to design MCQ. One possible way to do that is get the sentence and after defining the keyword, drop it and put a blank space instead. Also factoid questions can be used in multiple choice format as well. Therefore, we can use wh word for making MCQ.

In a system by [33] they have designed a procedure to generate multiple choice questions from electronic documents. They have occupied NLP techniques such as shallow parsing, automatic term extraction, sentence transformation and computation of semantic distance. In addition to those, their system used language resources such as corpora and ontologies. Finally, they provided an editor for the user, therefore, they can edit the output and make their changes to the questions. Following is an example of a question generated by their system:

```
What do words and phrases form?


a) the constituents of the clause.
b) the phrases of the clause.
c) the sentences of the clause.
d) the optional constituents of the clause.
```

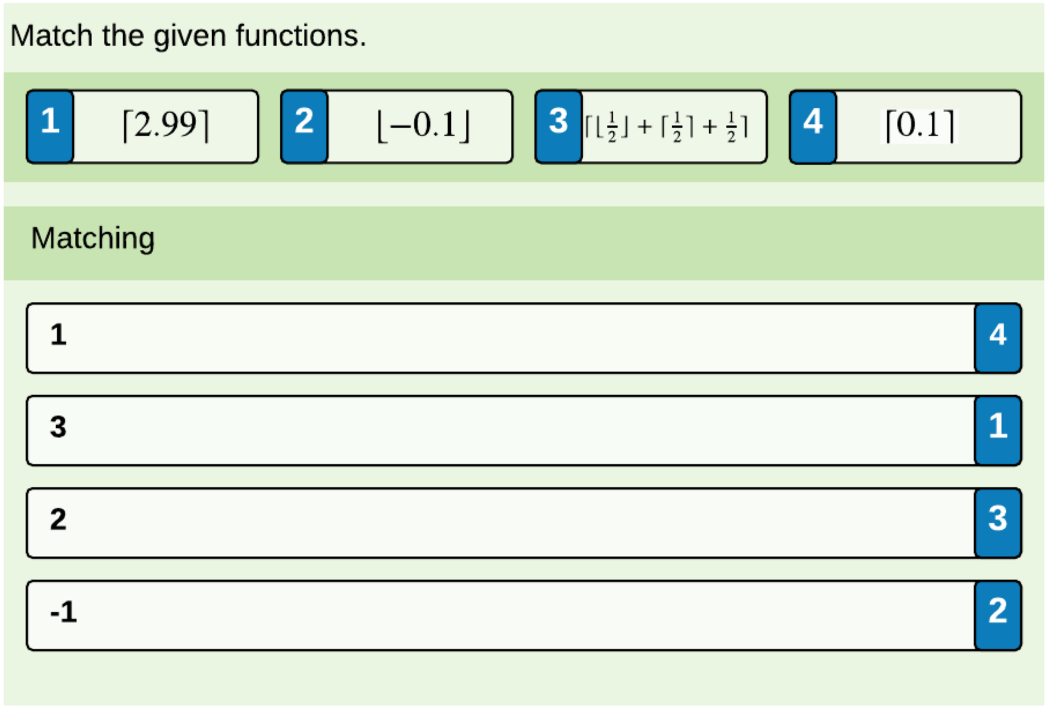
In another approach by Goto et al [17] they generated multiple choice questions in three steps. At the first, based on preference learning, they extracted applicable sentences from the input text . Second, by using a conditional random field they made the blank space. And Three, using statistical patterns of existing questions they developed distracters.

2.3 Question Generations Systems in Mathematics

Mathematics has always been one of the confusing topics for students. Therefore, it requires instructors to assign further exercises for learners. There are several question generations available for this field which utilize question banks. Although using question banks does not fit into automatic question generation category. However, by making some alterations in the output of these banks we can analyze them among automated question systems. The main strategy in these type of frameworks is storing limited number of predefined questions in a database, categorize them based on designated criteria and retrieve them when called. In this section we analyze some of the available question generation systems for mathematics. Unfortunately most of the available math question generators are business products which makes accessing their source code and algorithms impossible.

2.3.1 Systems that Utilize Question Banks

One of the popular systems which uses predefined questions is McGraw-Hill Connect system [2]. This system allows users to take tests for preferred courses as well as reviewing the content of related topic in the textbook. Also, at the end of each quiz, the system evaluates it and students can review their score. Another functionality of this system is displaying the performance of each student by utilizing a progress graph, calculating their average score and etc. These features make managing the classroom very easy for instructors. Figure 2.4 is an example template from Connect system. In this question five steps of solving a problem is given to the test taker. The user needs to put them in the right order by choosing numbers.


Match the given functions.

1 $\lceil 2.99 \rceil$ 2 $\lfloor -0.1 \rfloor$ 3 $\lceil \lfloor \frac{1}{2} \rfloor + \lceil \frac{1}{2} \rceil + \frac{1}{2} \rceil$ 4 $\lceil 0.1 \rceil$

Matching

1	4
3	1
2	3
-1	2

Figure 2.4: An Example Question from McGraw-Hill Connect.

Learnosity [1] is another question creation platform where the author can use its dynamic functions to create mathematical questions and generate tests. Although this system provides users with different tools to simplify generation of questions with an interactive user interface, however, questions need to be defined manually in order to be used in this system.

The advantage of using question banks is because these kind of questions are more focused on what students exactly learned in the classroom. They can follow Bloom's taxonomy more accurately than questions that are generated randomly.

In large question banks, picking the right question could be a challenge. In a study by Purohit [36], they designed an adaptive question bank management system that is intelligently picking questions from a rich database and representing the question model according to the inputs or parameters provided by the question paper designer. They used a concept map that is connected to the question database. This way they make sure that questions are generated based on a criteria such as Bloom's taxonomy. The result was a web based question paper generator. This system could also be a semi automatic question generation system that will be discussed.

2.3.2 Systems that Semi-automatically Generate Questions

With the advancements in software development and programming languages, specially mathematical libraries and modules, there are various options available for generating mathematical questions. Semi-automatic question generations are the systems that are hard-coded in some parts while using algorithms to automatically generated other parts of the question such as utilizing random numbers within a specific format that produces same question with different number each time. These type of question generations are used in most of the available systems. We discuss some of those systems in this section.

Mathsbot [19] is an interactive system for mathematics. Question generation is one of the features of this system, it has different types of tools for QG such as differentiated questions, loop cards, question generator, test maker, topic ladder and worksheet generator. It covers a variety of topics in mathematics including numbers, algebra, geometry, ratio and proportion, probability and statistics. One of the main issues with this system is lack of variety in questions for each topic. As an example, for probabilities, it covers expected frequency and simple events and for each of those topics there is only one question type available. The system changes the numbers with each request to make it semi-automatic.

Wolfram alpha [42] introduced in 2009 is a "search engine", but, what makes it different from other search engines, is its capability to answer the scientific questions. Problem generators are a part of this system that can produce different types of mathematical questions from topics such as algebra, calculus, statistics and number theory. In addition, Wolfram Alpha made its features available via an API that generates a universally available image format as a response for users' requests.

Chapter 3

Application Programming Interface (API)

3.1 API Technology

Application programming interfaces (APIs) are software intermediaries that allow multiple applications to interact with each other. Data exchange is performed between the client and the server by making calls and requests. Most of the companies and organizations such as startups, government entities, big tech companies are relying on their APIs systems for their backend operations [27]. The main feature that distinguishes APIs from websites is the user interface, this abstraction prevents the issues that are common with general websites [14]. APIs can represent data in different formats. The most common types of the data that APIs generate are JSON, XML and HTML.

3.2 REST

Representational state transfer or RESTful is a software architecture style that defines the way of exchanging information between web systems [9]. REST is a remote procedure call (RPC) protocol that utilizes HTTP protocol to transfer information. Another alternative RPC for REST is simple object access protocol or SOAP. However, simplicity, reliability, and ease of use of REST makes it a better choice. RESTful services let their users gain access and make alterations to web resources by utilizing an established set of stateless operations [11].

3.3 Client and Server

Calls in RESTful APIs are made between clients and servers. The server is defined as the host of the data and operations are made on the server-side. The server produces and obtains representations of resources. On the other hand, the client requests and receives the data from the server by making calls. The client is also responsible for manipulating and parsing the representations received by the server [14]. This procedure also makes REST API a client-server architecture. Figure 3.1 is a simple representation of an APIs architecture.

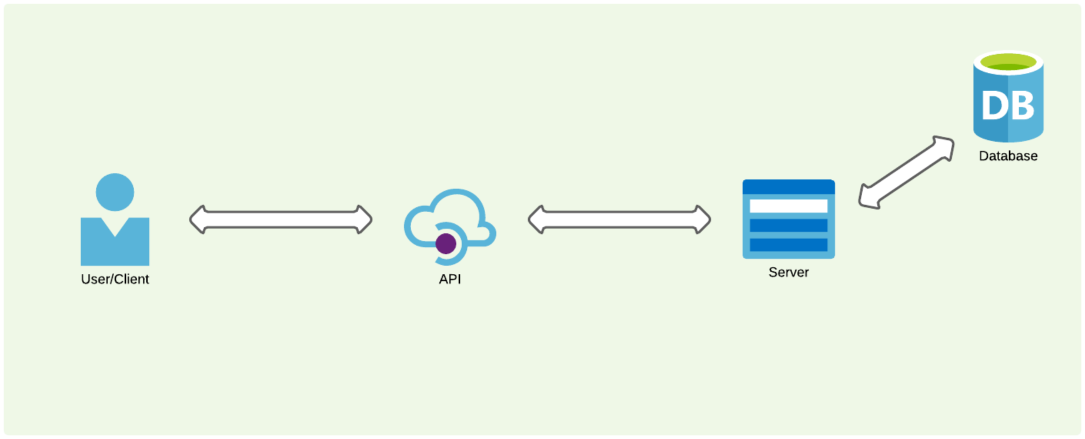


Figure 3.1: Architecture of a RESTful API.

3.4 HTTP

Hypertext transfer protocol(HTTP) is a protocol for hypermedia information systems [16]. It allows the exchange of resources between web applications [27]. HTTP has predefined methods that distinguish the type of request between two ends. The top four HTTP methods are GET, POST, PUT, and DELETE. The GET method requests a representation of the resources available from the server. The POST method sends a resource to the server. The PUT method updates an existing resource and the DELETE method removes a specific resource. After each request, the server issues a specific code that indicates the response from the server to the client's request which is called the status code. Table 3.1 shows the different status codes for GET, POST, PUT, and DELETE.

Status Code	Description
200	OK
201	Created
202	Accepted
301	Moved Permanently
303	See Other
304	Not Modified
307	Temporary Redirect
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
410	Gone
500	Internal Server Error
503	Service Unavailable

Most frequent HTTP status codes and their descriptions. After each HTTP request, the server issues a specific code that indicates the response from the server to the clients request.

Table 3.1: HTTP Status Codes.

3.5 JSON

JavaScript object notation or JSON is a text data format that is similar to the objects in JavaScript or dictionaries in Python, however, it is an independent file format. JSON can include a variety of data types such as numbers, strings, arrays or lists, objects, booleans, and null. One of the most used formats to represent information in APIs is by using JSON. Following is an example of a basic JSON file.

```
{"name":"John",
"lastName":"Doe",
"Age":25,
"classes":[
    {
    "classID":"CSCI1",
    "ClassName":"Introduction to Algorithms"},
    {
    "classID":"CSCI2",
    "ClassName":"Discrete Mathematics"
    }
    ]}
```

Chapter 4

Question Generation API Development

4.1 Proof of Concept

Questions are indisputably essential elements in learning. They are crucial for gaining and assessing knowledge. Asking and answering questions helps to extract the information from memory, clarify the misconceptions, reveal the important pieces of a concept, and allows the learner to repeat the core materials. Questions may be useful both before and after the learning process as they attract the learners' attention before knowledge is gained by emphasizing the core concepts of the topic. Furthermore, questions help the learner to retrieve the information and receive proper feedback regarding the material that they retain following the learning process. Discrete mathematics is one of the most critical and complex courses for computer science students. It takes more practice to completely understand the concepts and implement them in solving problems. Therefore, instructors need to provide more tools for their students to assess their knowledge, however, creating and providing proper components is a time-consuming activity. Let's say there is a hypothetical class of 90 students with an instructor that assigns a practice quiz with only 5 questions for each student in order to determine their comprehension. It would be 450 questions and evaluating all those questions would take a lots of time. Despite this fact, the necessity for an automated system that can perform the question asking and evaluating stage of the

learning process is unavoidable. The idea of having a system that can automatically generate questions for different topics and also provide the answer for them would be a tremendous asset for both instructor and learner alike. The recent advancements in web services are encouraging the developers to create products that run on web servers which are called web applications. Therefore, having a system that can produce questions and remain easily accessible for future developers to implement into their systems made us execute our idea as an application programming interface or API. The proposed system will be developed in Python using Flask as the main framework. Multiple functions are going to be constructed, some of them are directly generating questions while others are responsible for local operations.
Figure 4.1 shows the proposed file structure of the system. After the execution of functions.py which is responsible for our question generation algorithms, the output will be sent to the jsonify.py which retrieves the required data and returns them in JSON format. App.py will be responsible for server configuration and path definition.

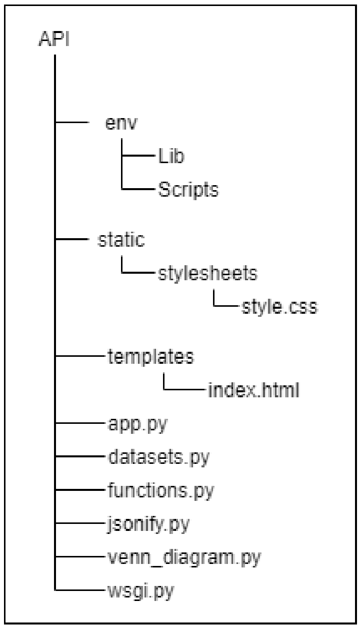


Figure 4.1: File Structure of The System.

It is going to run on a server to make it available for users. The system will implement both required and optional arguments. The main required argument will be the main topics such as set-union, which users must consider in order to make calls. There will be multiple optional arguments such as number of questions and type of elements used in each set for the question. Finally the results will be returned to the user in JSON format. Figure 4.2 represents the high level architecture of the API.

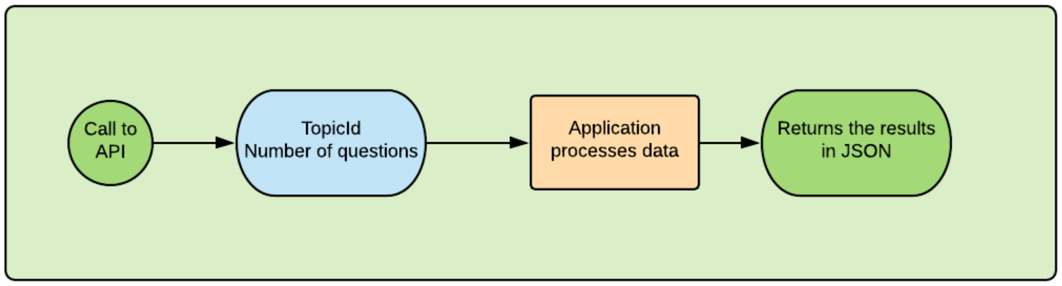


Figure 4.2: High Level Architecture of the Question Generation System.

4.2 Design and Development of QG API Using Flask

4.2.1 Requirements

We defined 8 requirements that our system should meet after the design. Table ... shows the list of all requirements.

Requirement No.	Requirement Description
R1	The system should be running on a server.
R2	User should be able to customize parameters such as question numbers.
R3	User should be able to generate question by topic id.
R4	A unique id should be assigned to each question.
R5	Common errors should be predicted and handled properly.
R6	Equations should be generated as LaTex and Mathjax format.
R7	Correct answer should be provided for each question.
R8	The output should be in JSON format.

List of requirements for the question generation system.

Table 4.1: Requirements List.

4.2.2 Proposed Framework

In order to meet our requirements, we can use Python and Flask as our main frameworks. By default, Python has a variety of mathematical functions such as set operations. We will use the “random” library to generate random integers and floats and select items from datasets. For drawing diagrams such as Venn diagrams, we are going to use matplotlib-venn [39], and “base64” [12] for encoding the output diagram and use it in string format in the API. For assigning a unique id for each question "uuid"[1] library will be used. It's a library that generates 128-bit numbers that is universally unique and can help to distinguish each question for future use. We will wrap up output in the Python dictionary which will be converted to a JSON file. The framework will be tested using Postman.

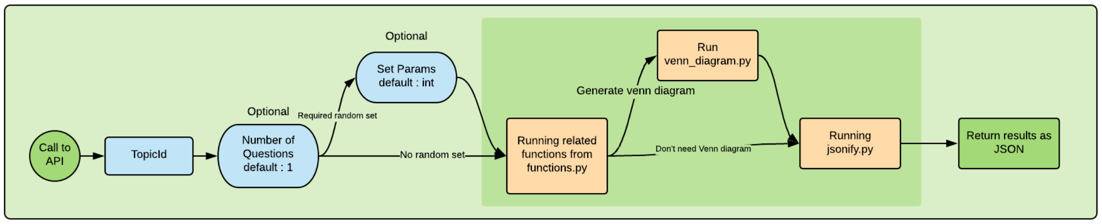


Figure 4.3: Architecture of the API System for Automatically Generated Questions.

4.2.3 Python

Python is a high-level, interpreted programming language, it is compared to executable pseudocode because of its simple syntax [34]. Python was created and released by Guido van Rossum in 1991 it focuses on code abstraction and readability [25]. Many factors made Python one of the popular programming languages. It is open source which lets users to create and distribute their product easily, the coherence of the syntax not only helps to reduce the lines of the code and increase productivity of developers but also makes it easier to maintain and even extend. One of major advantages of Python is the large pre-installed libraries called standard library which covers majority of development tasks. In addition to that, extra libraries are available and easily installable which increases the efficiency. The interactive interpreter of Python improves the process of development and testing [34].

[1]https://pypi.org/project/uuid/

4.2.4 Flask

Flask is a web framework also called a micro-framework written in Python. It consists of two main components, Werkzueg which is responsible for routing, debugging and web server gateway interface (WSGI) applications, and Jinja2 which is responsible for templating [18]. Flask is a simple and flexible framework which makes it easy to use for developing a web server. We use Flask as our main framework to create server for our framework and also handle clients' requests.

4.2.5 Python Environment Setup

We use a virtual environment to store our packages and dependencies. It is a tool that makes a self-contained directory tree using a specific version of Python, this way we can install our packages in a particular directory without affecting the global Python interpreter. It also helps to prevent version conflict between local and global interpreters. We create our virtual environment using the "virtualenv venv" command. Then we can activate our virtual environment by using scripts that are provided by virtualenv and start adding dependencies. Table 4.2 represents all the libraries and their versions used in our system.

Library Name	Version
Flask	1.0.2
Flask-Cors	3.0.7
Flask-RESTful	0.3.7
Flask-SQLAlchemy	2.4.1
Werkzeug	0.14.1
gunicorn	19.9.0
Click	7.0
itsdangerous	1.1.0
Jinja2	2.10.1
MarkupSafe	1.1.1
uuid	1.30
pybase64	1.0.1
matplotlib-venn	0.11.5
matplotlib	3.3.2

List of all the Python libraries utilized in the system with their versions. All the dependencies should be installed in order to use the system properly.

Table 4.2: List of Libraries Used in the Framework.

We host the server on our local machine. Below is the Python code for running the server.

```
if __name__ == '__main__':
    app.debug = True
    app.run(host='127.0.0.1', port=5000)
```

4.2.6 Setting up Routings

For the API to respond to the clients requests, we set up routings in a consistent format. Therefore, each question is accessible by its specific path. These paths are being defined by decorators in Flask. Decorators are useful tools in Python that allows developers to modify the behaviour of classes and functions, they are defined with an at sign(@). Next we define the routing path and specify the allowed methods for it. Following snippet from Python is an example of defining a routing for the index page of the system.

```
@app.route('/', methods=['GET'])
```

4.2.7 Mathematical Algorithms

The core component of this study is mathematical algorithms that are created to generate questions. We developed 51 algorithms for four major topics in discrete mathematics, 35 algorithms are directly generating questions while 16 algorithms are responsible for different processes. For classification purposes, topic IDs are assigned to each algorithm that is responsible for each question. Algorithms are composed as Python functions. In this section we discuss different topics used in our system.

Set Theory

A set is defined as a collection of objects. Almost all objects in mathematics are sets. Therefore, we can define set theory as a foundation of mathematics. Set theory was the first topic we developed in the system. The topics are covered by the framework are set operations, partition of a set and Venn diagram. Table 4.3 is the list of all topic ids and their end-point URLs provided in the API for set theory.

Topic ID	Topic Name	API End-Point URL
ds:set-theory:set-union	set-union	GET /set-union
ds:set-theory:set-difference	set-difference	GET /set-difference
ds:set-theory:set-intersection	set-intersection	GET /set-intersection
ds:set-theory:set-symmetric-difference	set-symmetric-difference	GET /set-symmetric-difference
ds:set-theory:cartesian-product	cartesian-product	GET /cartesian-product
ds:set-theory:venn-diagram	venn-diagram	GET /venn-diagram
ds:set-theory:power-set	power-set	GET /power-set
ds:set-theory:set-partition	set-partition	GET /set-partition

List of all end-points for topics in set theory. Using these end-point URLs, calls can be made to the system and questions can be generated for each topic.

Table 4.3: API End-Point URLs for Set Theory.

Random Set

The first step in making a question for set theory is creating a set, for that reason, we need to have a function that can randomly generates sets with different parameters so that users can define those parameters. Table 4.4 shows the list of all available parameters to customize set items. As default, with no parameters given, it will return a set with 5 random integers ranging from 1 to 20 for each set.

Parameter	Definition
integer	Number of integers in the set.
float	Number of floats in the set.
char	Number of characters in the set.
country_name	Number of country names in the set.
city_name	Number of city names in the set.
male_name	Number of male names in the set.
female_name	Number of female names in the set.
integer_min	Minimum value of integer to choose from.
integer_max	Maximum value of integer to choose from.
float_min	Minimum value of float to choose from.
float_max	Maximum value of float to choose from.
float_dec	Number of decimal points for each float.
integer_type	Even, odd or mix.
heterogeneous	Whether duplicates are allowed in the set or not.

List of parameters that define item types in each random set, by using these parameters when calling the function we can customize the elements in each set.

Table 4.4: List of Parameters for Random Set Generator Function.

Set Operations

Set operations play a key role in set theory. By utilizing set operations we are creating new sets by combining two or more sets. The operations used in this system are the union of sets, the intersection of sets, the difference of sets, the cartesian product, the complement of sets, and the symmetric difference of sets. All the set operations are being made by a function named set_operations. There are three parameters by which users can customize this function. The first one is "op" that defines the type of operation to perform such as the union of sets, and two others are set_1, and set_2 that define input sets for the chosen operation. As an example for getting a union of two sets that each has three integers we can use:

```
set_operation(op='set-union', random_set(integer=3),
    random_set(integer=3))
```

Venn diagram

Introduced by John Venn in 1880 [40], Venn diagrams are used to represent all possible relations between sets. Venn diagrams use curves to define the boundaries between sets and each element is placed in the region it belongs. These curves could overlap if there are any common elements between their correspondent sets. Matploptlib-venn [39] is a library that is created using matplotlib [21] in Python to draw Venn diagrams with two or three circles. We use this dependency to plot Venn diagrams for some questions in set theory. The file named venn_diagram.py is where we defined function venn2() that takes two sets and returns a Venn diagram representation of them. Figure 4.4 is a simple Venn diagram generated using two sets of A={5,6,7,9,14,17,19} and B={2,3,6,11,14,16,17,21,26}.

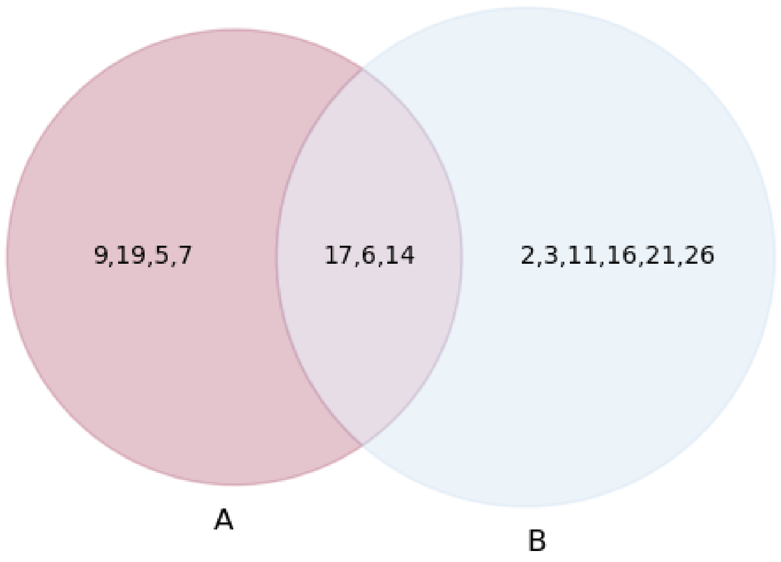


Figure 4.4: An Axample of a Venn Diagram for Two Sets of A={5,6,7,9,14,17,19} and B={2,3,6,11,14,16,17,21,26}

Functions

Functions are defined as a relation between two sets that associates elements of the first set to an element from the second set. Without any doubt, functions play an important role in mathematics and computer science. Each function has a domain and target. Domain is all the possible inputs for a function and target or codomain is a set of all outputs of a function given a particular set of inputs.

In our system, we are using programming features to generate questions for function definition, different types of functions such as surjective, injective, and bijective, floor function, ceiling function, the inverse of a function, domain of a function and target of a function. Table 4.5 displays topic ids and API end-point URLs for each topic in functions.

Topic ID	Topic Name	API End-Point URL
ds:functions:function-definition	function-definition	GET /function-definition
ds:functions:function-definition	one-to-one-function	GET/one-to-one-function
ds:functions:inverse-function	inverse-function	GET /inverse-function
ds:functions:floor-function	floor-function	GET /floor-function
ds:functions:ceiling-function	ceiling-function	GET /ceiling-function
ds:functions:function-domain	function-domain	GET /function-domain
ds:functions:function-target	function-target	GET /function-target

List of all end-points for topics in functions. Using these end-point URLs, calls can be made to the system and questions can be generated for each topic.

Table 4.5: API End-Point URLs for Functions.

Making Relations

Defining relations in Python was the main challenge for making questions for function. We needed to make two sets to represent domain and codomain. For this purpose, we created an algorithm to generate two sets with 3 to 6 elements in each and the range of each set is chosen from 1 to 10 randomly. Zip function is a built-in Python function that takes iterables as an input, pairs them in tuples, and returns them. In the example below we used zip function to make relations between two sets of A and B.

```
A = {1, 2, 3}
```

```
B = {4, 5, 6}
output = set(zip(A, B))
```

The output was as bellow:

```
{(2, 5), (3, 6), (1, 4)}
```

Equations

Equations are important part of mathematics, therefore, generating mathematical questions without the ability of rendering equations would result in various limitations to the system. We used LaTeX statements in order to generate questions with mathematical equations. Although our system contains only one question for one-to-one function that uses equations. However, by adding that feature we can extend our framework to cover new topics such as algebra in the future. For rendering equations we can use MathJax [8]. MathJax is a JavaScript library for displaying mathematical equation by utilizing LaTex and ASCIIMathML markup. It can be used by wrapping LaTex between double dollar signs. Figure 4.5 displays the output of rendering code below using Mathjax.

```
$${\sqrt{x} \over 2}$$
```

$$\frac{\sqrt{x}}{2}$$

Figure 4.5: Mathjax Rendering of `$${\sqrt{x} \over 2}$$`

Probabilities

Probability in mathematics is predicting the likelihood of occurrence of an event using mathematical functions. The probability of an event is presented by a number between 0 and 1 which the closer the number gets to 0 means the event is unlikely to happen and vice versa. Probability is an important and confusing component of discrete mathematics. Frequent practicing is the key to understand the concepts and use them in problem-solving. Therefore, we chose it as one of our topics for this system.

We are covering different topics in probability. Table 4.6 is all the available topics and their path in the API. For some topics we generated more than one algorithm, as an example for multiplication, there are 4 different types of questions that are randomly being used when called. Python's built-in math library is used in some of the questions for calculating factorials.

Topic ID	Topic Name	API End-Point URL
ds:probabilities:event-probability	event-probability	GET /event-probability
ds:probabilities:permutation	permutation	GET /permutation
ds:probabilities:multiplication-rule	multiplication-rule	GET /multiplication-rule
ds:probabilities:combination	combination	GET /combination
ds:probabilities:conditional-probability	conditional-probability	GET /conditional-probability
ds:probabilities:probability-union	probability-union	GET /probability-union
ds:probabilities:probability-complement	probability-complement	GET /probability-complement
ds:probabilities:bayes-theorem	bayes-theorem	GET /bayes-theorem

List of all end-points for topics in probabilities. Using these end-point URLs, calls can be made to the system and questions can be generated for each topic.

Table 4.6: API End-Point URLs for Probabilities.

Relations

A relation or a binary relation over two sets in mathematics is referred to a relationship between the x values and y values of ordered pairs. The set of all x values is called the domain, and the set of all y values is called the range. Values in the domain should be unique. Figure 4.6 is displaying a relation between two sets of numbers. We can define domain, target and ordered pairs as below:

$$ordered\ pair = \{(2,-3),(5,0),(8,0),(7,9),(10,4)\}$$

$$domain = \{2,5,7,8,10\}$$

$$target = \{-3,0,4,9\}$$

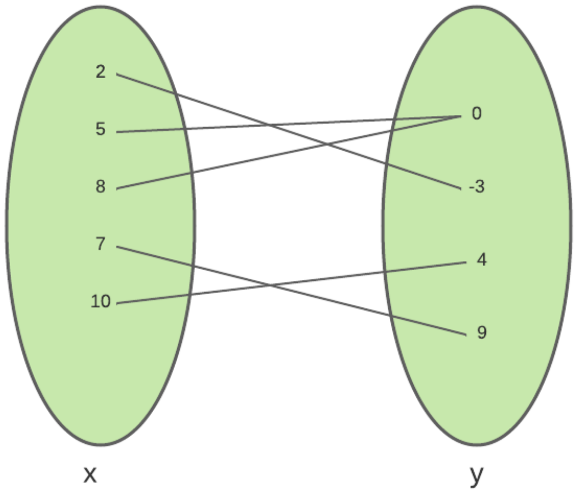


Figure 4.6: Displaying a Relationship as Mapping.

The topics covered in this system for relations are reflexive relation, irreflexive relation, transitive relation, asymmetric relation, anti symmetric relation, reflexive closure, symmetric closure and transitive closure. We used booleans to generate true or false questions for these topics. Table 4.7 is the list of topic ids and their end-point URLs in the API for relations.

Topic ID	Topic Name	API End-Point URL
ds:relations:reflexive-relation	reflexive-relation	GET /reflexive-relation
ds:relations:irreflexive-relation	irreflexive-relation	GET /irreflexive-relation
ds:relations:symmetric-relation	symmetric-relation	GET /symmetric-relation
ds:relations:asymmetric-relation	asymmetric-relation	GET /asymmetric-relation
ds:relations:antisymmetric-relation	antisymmetric-relation	GET /antisymmetric-relation
ds:relations:transitive-relation	transitive-relation	GET /transitive-relation
ds:relations:reflexive-closure	reflexive-closure	GET /reflexive-closure
ds:relations:symmetric-closure	symmetric-closure	GET /symmetric-closure
ds:relations:transitive-closure	transitive-closure	GET /transitive-closure

List of all end-points for topics in relations. Using these end-point URLs, calls can be made to the system and questions can be generated for each topic.

Table 4.7: API End-Point URLs for Relations.

4.2.8 Getting Output as JSON

As stated in 4.2.2, JSON is selected as our output format. For that reason, we created a function to wrap all the outputs in a unified JSON format, therefore, we could have consistency in the representation of all the questions. The output JSON has different parts. Below is the JSON format used in our system as an output:

```
{
 "questions": [
  {
    "answerSelectionType": "single",
    "answers": list
    "correctAnswer": int, String or Boolean
    "difficulty": int,
    "messageForCorrectAnswer": "CORRECT ANSWER",
    "messageForIncorrectAnswer": "INCORRECT ANSWER",
    "point": int,
    "question": String,
    "questionID": String,
    "questionType": String
  }
 ],
 "quizSynopsis": String,
 "quizTitle": String
}
```

As stated in 4.2.7, users can customize the output by adding some arguments to the call. For all topics the user can define the number of the question to be generated, additionally, for set theory questions the user can also define parameters that define the elements in each of the sets. As an example for requesting a question for a set union with 5 questions with the first set containing integers and second set containing float numbers, the following URL is used. Table 4.8 displays the available parameter for making the calls.

```
GET /set-union/5/12
```

Parameter	Element Type
1	Integer
2	Float
3	Character
4	Country Name
5	City Name
6	Male Name
7	Female Name

List of parameters that define item types in each set. Users can use these parameters to make customized sets for their questions. These parameters should be used in the end-point URL. As an example /set-union/1/11 when we want integers in both sets.

Table 4.8: Set Parameters in End-Point URL.

Chapter 5

Evaluation

There are different methods and techniques to evaluate an API, we test our system using Postman[1] and a Python script. With Postman we evaluate the response of the system when we call each topic and error handlings when an erroneous request happens. Furthermore, with our Python script we send multiple requests using different parameters to assess our systems responses for each call.

5.1 Testing using Postman

Postman is a development environment for APIs. It has a robust collection of built-in resources to help developers evaluate and improve their APIs. Postman has the ability to run multiple cases with a single test, therefore, our first test is sending GET requests to each of our end-points with no parameters and making sure that we are getting an OK (200 status code) response. Additionally, the test should indicate whether the server is returning any content or not, if so the response is in JSON format. In order to perform our first test 32 requests each with 10 iterations sent to the API and following script used to evaluate the response.

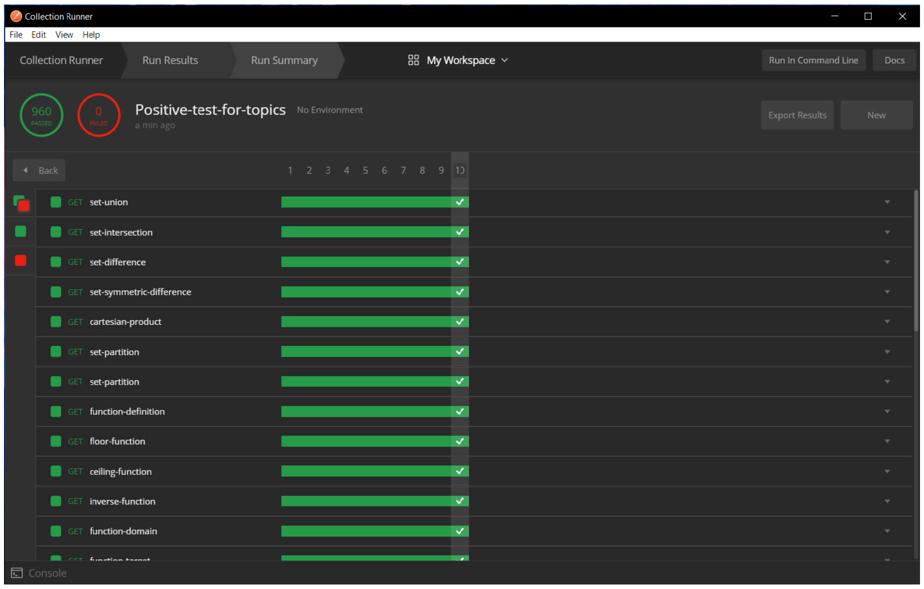
```

// Checking status code
```

[1] www.postman.com

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});

// Checking existence of content-type
var contentTypeHeaderExists = responseHeaders.hasOwnProperty(
   "Content-Type");

tests["Has Content-Type"] = contentTypeHeaderExists;

// Checking to see if content-type is json
if (contentTypeHeaderExists) {
    tests["Content-Type is application/json"] =
      responseHeaders["Content-Type"].has("application/json")
         ;
}
```

By using the script above in Postman, as shown in figure 5.1 we conducted 960 tests for 32 test cases and got 960 successful results.

Consequently, we conducted another test with Postman to evaluate the functionality of event handlers. We planned 16 different scenarios that users can face an error including but not limited to sending a request that is not allowed(status code 405), requesting a path which does not exist(status code 404) and server error(status code 500). Each test scenario is using different scripts to check the proper status code, response body and format of the response body. Each request executed 10 times and with a total of 460 tests, we got 460 successful responses for each error as shown in figure 5.2.

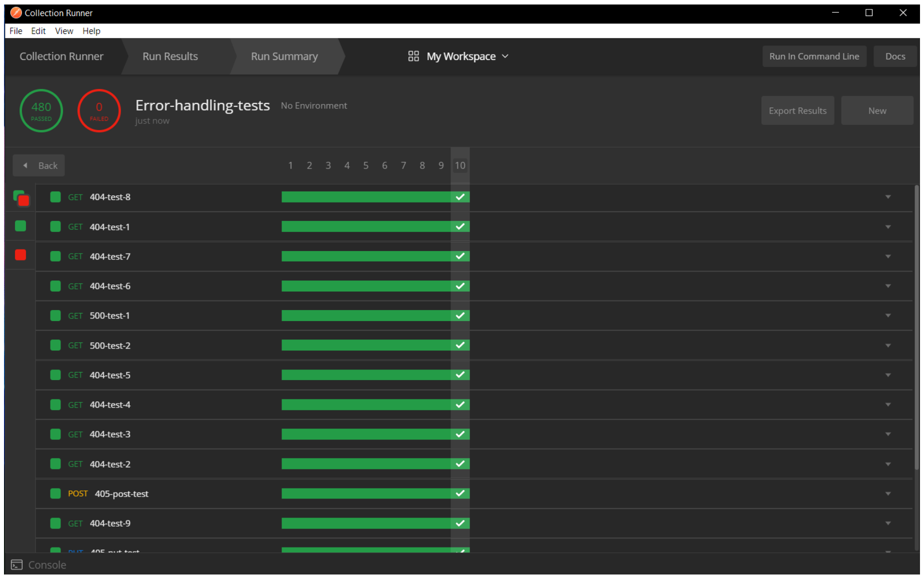


960 PASSED 0 FAILED

Positive-test-for-topics No Environment
a min ago

GET set-union
GET set-intersection
GET set-difference
GET set-symmetric-difference
GET cartesian-product
GET set-partition
GET set-partition
GET function-definition
GET floor-function
GET ceiling-function
GET inverse-function
GET function-domain

Figure 5.1: Test Results for Topics with no Parameters in Postman.



480 PASSED 0 FAILED

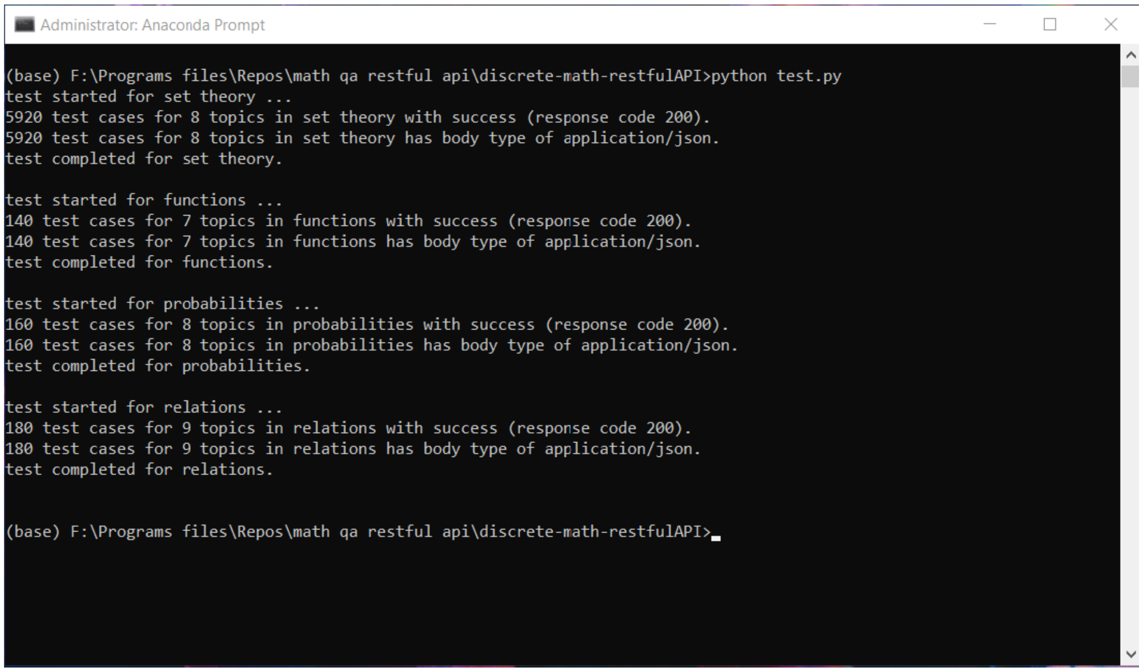
Error-handling-tests No Environment
just now

GET 404-test-8
GET 404-test-1
GET 404-test-7
GET 404-test-6
GET 500-test-1
GET 500-test-2
GET 404-test-5
GET 404-test-4
GET 404-test-3
GET 404-test-2
POST 405-post-test
GET 404-test-9

Figure 5.2: Test Results for Error Handlers in Postman.

5.2 Testing using Python Script

Finally we created a script using Python to make multiple calls for each end-point of our API using different parameters, for each specific parameter with requests from 1 to 20 questions. As a result we executed 5920 tests for set theory, 140 tests for function, 160 tests for probabilities and 180 tests for relations. In this test we made sure that our API sent an OK response (status code 200) and the response was in application/json format for each call. Figure 5.3 is displaying the test results.



```
(base) F:\Programs files\Repos\math qa restful api\discrete-math-restfulAPI>python test.py
test started for set theory ...
5920 test cases for 8 topics in set theory with success (response code 200).
5920 test cases for 8 topics in set theory has body type of application/json.
test completed for set theory.

test started for functions ...
140 test cases for 7 topics in functions with success (response code 200).
140 test cases for 7 topics in functions has body type of application/json.
test completed for functions.

test started for probabilities ...
160 test cases for 8 topics in probabilities with success (response code 200).
160 test cases for 8 topics in probabilities has body type of application/json.
test completed for probabilities.

test started for relations ...
180 test cases for 9 topics in relations with success (response code 200).
180 test cases for 9 topics in relations has body type of application/json.
test completed for relations.


(base) F:\Programs files\Repos\math qa restful api\discrete-math-restfulAPI>
```

Figure 5.3: Test Results for Topics with Parameters Using Python Script.

Chapter 6

Future Work and Conclusion

6.1 Future Work

As stated in Chapter 2, automatic question generation systems are mostly available for natural languages and extracting questions from texts. In a novel approach in this research we developed an automatic question generation for discrete mathematics, however, we only covered four main topics, future researchers and developers could generate questions using similar techniques for other topics that are not covered in this framework. Furthermore, questions being generated by our system are multiple choice or true or false questions, therefore, adding more types of questions such as fill in the blank questions could be considered for future developments.

With current system, user can only make GET requests and questions can only be modified on the source code, hence, another possible extension for our system could be integrating the framework with a database, this way manually generated questions can be added to the database and retrieved when needed and other HTTP methods that are discussed in chapter 3 such as POST, PUT and DELETE can be utilized in the API.

Available automatic question generations are mainly focused on a single topic, this makes it hard to generate a quiz or test automatically. Developing a system that can use these algorithms and generates a test for a given topic or even a given course could be a tremendous help for instructors and students.

6.2 Conclusion

In this research, after analyzing existing systems for auto question generation for both natural languages and mathematics, we presented our framework for automatically generating questions from different topics in discrete mathematics. Our system was able to generate questions for set theory, functions, probabilities and relations. Our API was able to generate questions based on parameters that the user defined and output them as JSON files. We believe by utilizing this system students would be able to improve their knowledge by continuously practicing without any need for their instructor to create them questions, and also instructors could save time and effort in making questions for their students.

BIBLIOGRAPHY

[1] Learnosity. `https://authorguide.learnosity.com/hc/en-us`.

[2] McGraw Hill Connect. `https://www.mheducation.com/highered/support/connect.html`.

[3] Adamson, D., Bhartiya, D., Gujral, B., Kedia, R., Singh, A., and Rosé, C. P. Automatically generating discussion questions. In *International Conference on Artificial Intelligence in Education* (2013), Springer, pp. 81–90.

[4] Agarwal, M., and Mannem, P. Automatic gap-fill question generation from text books. In *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications* (2011), Association for Computational Linguistics, pp. 56–64.

[5] Agarwal, M., Shah, R., and Mannem, P. Automatic question generation using discourse cues. In *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications* (Portland, OR, 2011), Association for Computational Linguistics, pp. 1–9.

[6] Ali, H., Chali, Y., and Hasan, S. A. Automation of question generation from sentences. In *Proceedings of QG2010: The Third Workshop on Question Generation* (2010), pp. 58–67.

[7] Bloom, B. S., Englehart, M. D., Furst, E. J., Hill, W. H., and Krathwohl, D. R. Taxonomy of educational objectives. vol. 1: Cognitive domain. *New York: McKay* (1956), 20–24.

[8] Cervone, D. Mathjax: a platform for mathematics on the web. *Notices of the AMS 59*, 2 (2012), 312–316.

[9] Chen, X., Ji, Z., Fan, Y., and Zhan, Y. Restful api architecture based on laravel framework. In *Journal of Physics: Conference Series* (2017), vol. 910, IOP Publishing, p. 012016.

[10] Cisar, S. M., Cisar, P., and Pinter, R. True/false questions analysis using computerized certainty-based marking tests. In *2009 7th International Symposium on Intelligent Systems and Informatics* (2009), IEEE, pp. 171–174.

[11] Consortium, W. W. W., et al. Relationship to the world wide web and rest architectures, 2004.

[12] Darbois, M. Fast Base64 stream encoder/decoder. `github.com/mayeut/pybase64`, Feb. 2015.

[13] Divate, M., and Salgaonkar, A. Automatic question generation approaches and evaluation techniques. *Current Science 113*, 9 (2017), 1683.

[14] Eizinger, T. *API design in distributed systems: a comparison between GraphQL and REST*. PhD thesis, Master's Thesis). University of Applied Sciences Technikum Wien-Degree, 2017.

[15] Eom, J.-S., Lim, U.-K., and Lee, J.-Y. Automatic generation system of true or false questions. *Advanced Science and Technology Letters 135* (2016), 46–49.

[16] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. Hypertext transfer protocol–http/1.1, 1999.

[17] Goto, T., Kojiri, T., Watanabe, T., Iwata, T., and Yamada, T. Automatic generation system of multiple-choice cloze questions and its evaluation. *Knowledge Management & E-Learning 2*, 3 (2010), 210–224.

[18] Grinberg, M. *Flask web development: developing web applications with python.* " O'Reilly Media, Inc.", 2018.

[19] Hall, J. Question generator. `mathsbot.com/questionGenerator`.

[20] Heilman, M., and Smith, N. A. Extracting simplified statements for factual question generation. *Third Workshop on Question Generation, Tenth International Conference on Intelligent Tutoring Systems ITS2010* (2010).

[21] Hunter, J. D. Matplotlib: A 2d graphics environment. *Computing in Science Engineering 9*, 3 (2007), 90–95.

[22] Kalady, S., Elikkottil, A., and Das, R. Natural language question generation using syntax and keywords. In *Proceedings of QG2010: The Third Workshop on Question Generation* (2010), vol. 2, questiongeneration. org, pp. 376 – 385.

[23] Khullar, P., Rachna, K., Hase, M., and Shrivastava, M. Automatic question generation using relative pronouns and adverbs. In *Proceedings of ACL 2018, Student Research Workshop* (2018), pp. 153–158.

[24] Krathwohl, D. R., and Anderson, L. W. *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives.* Longman, 2009.

[25] Kuhlman, D. *A python book: Beginning python, advanced python, and python exercises.* Dave Kuhlman Lutz, 2009.

[26] Kumar, G., Banchs, R., and D'Haro, L. F. RevUP: Automatic gap-fill question generation from educational texts. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications* (2015), pp. 154–161.

[27] Lauret, A. *The Design of Web APIs.* Manning Publications, 2019.

[28] Levy, R., and Andrew, G. Tregex and Tsurgeon: tools for querying and manipulating tree data structures. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)* (2006), pp. 2231–2234.

[29] Liu, M., and Calvo, R. A. An automatic question generation tool for supporting sourcing and integration in students' essays. *ADCS 2009* (2009), 90.

[30] Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., and McClosky, D. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations* (2014), pp. 55–60.

[31] Mazidi, K., and Nielsen, R. D. Leveraging multiple views of text for automatic question generation. In *International Conference on Artificial Intelligence in Education* (2015), Springer, pp. 257–266.

[32] Miller, G. A. WordNet: a lexical database for English. *Communications of the ACM 38*, 11 (1995), 39–41.

[33] Mitkov, R., Le An, H., and Karamanis, N. A computer-aided environment for generating multiple-choice test items. *Natural language engineering 12*, 2 (2006), 177–194.

[34] Oliphant, T. E. Python for scientific computing. *Computing in Science & Engineering 9*, 3 (2007), 10–20.

[35] Piwek, P., and Stoyanchev, S. Question generation in the CODA project. In *Proceedings of QG2010: The Third Workshop on Question Generation* (2010), pp. 58–67.

[36] Purohit, V. K., Kumar, A., Jabeen, A., Srivastava, S., Goudar, R., Rao, S., and Shivanagowda. Design of adaptive question bank development and management system. In *2012 2nd IEEE International Conference on Parallel, Distributed and Grid Computing* (2012), IEEE, pp. 256–261.

[37] Scott, T. Bloom's taxonomy applied to testing in computer science classes. *Journal of Computing Sciences in Colleges 19*, 1 (2003), 267–274.

[38] Thalheimer, W. The learning benefits of questions. *Work Learning Research* (2003).

[39] Tretyakov, K. Area-weighted venn-diagrams for Python/matplotlib. `github.com/konstantint/matplotlib-venn`, Oct. 2017.

[40] Venn, J. I. on the diagrammatic and mechanical representation of propositions and reasonings. *The London, Edinburgh, and Dublin philosophical magazine and journal of science 10*, 59 (1880), 1–18.

[41] Wolfe, J. H. Automatic question generation from text-an aid to independent study. In *ACM SIGCUE Outlook* (1976), vol. 10, ACM, pp. 104–112.

[42] Wolfram. Wolfram: Alpha: Making the world's knowledge computable. `https://www.wolframalpha.com/`.