Procedural Content Generation for Math Education

By
Yi Xu

Procedural Content Generation for Math Education

Submitted in fulfilment of the requirements for the degree of
Master of Science
In
Computer Science

Committee member:
Supervisor: Dr. ir. Rafael Bidarra
Member: Prof. Dr. Elmar Eisemann
Member: Dr. Christoph Lofi

Computer Graphics and Visualizations Group
Faculty of
Electrical Engineering, Mathematics and Computer Science
Delft, The Netherlands

Abstract

Mathematics education plays an essential role in children's development. In the past few years, online mathematics learning has gained increasing popularity. The online learning platform needs a large variety of textual and visual content to offer children a convenient learning experience and help them practice various mathematical skills. However, manually creating content is hugely time-consuming, expensive, and tedious for the content editors.

This project proposes a generic approach for procedural generation of mathematical problems and corresponding textual and visual content. We analyzed and clustered hundreds of primary school curriculum-based math knowledge components, and built flexible templates for generating abstract math problems, including arithmetic, comparison, ordering, mathematical relationships, measurements, and geometry. Then our system realizes the abstract math problems in natural language through the lexicalization of language-independent semantic configurations and syntactically structured templates. Our system generates visual content through text-based image retrieval and visualization of abstract math content, varying in the forms of table, chart, geometry, or picture for counting objects. Human expert evaluations found that our generated contents are understandable, sensible, and achieve well usefulness for primary school students.

Preface

This project is a part of my Master Thesis for the degree in Computer Science at the Delft University of Technology. The topic of the project is the Procedural Content Generation, and I found it very interesting for exploring this state-of-the-art research field.

This thesis project is being done with Squla, an international company in the Netherlands, providing an interactive, fun educational learning platform. I want to thank Roger Smeets, my mentor in sQula, who helped me solve problems and conduct the evaluation. I want to thank Pauline Rambonnet and Peter Hofstede for their advice and feedback on the content generation pipeline designing.

I would like to express my heartfelt thanks to my supervisor, dr.ir.Rafael Bidarra for suggesting this research topic to me and helping me find directions and solutions to the problems that happened during the project, and thanks for his time, patience, and valuable advice on our weekly discussion.

Lastly, I want to thank my parents and friends for their company and encouragement remotely during this work-from-home period.

Contents

List of Tables

List of Figures

1

Introduction

1.1 Background

Math education is an essential part of children's development. Mathematics provides opportunities for developing vital intellectual skills in problem-solving, deductive and inductive reasoning, logical and creative thinking, and communication. With the development of computer science and digital communication, online math education has attracted increasing attention. Common activities in the online learning environment include browsing pages, watching videos, viewing documents, answering online quizzes or tests, posting in forums, submitting homework, and peer evaluation. Among all these activities, answering online quizzes is the most widespread and widely-used one. It generally consists of a set of questions in the forms like multiple-choices and fill-in-the-blanks. Because the answers to multiple-choice questions and fill-in-the-blank questions are predefined, students can directly get corresponding feedback and scores from the platform, instantly after submitting the answers. By recording users' quizzing data, such as the answers they submitted, their time spent on each quiz, and their attempted answers to the questions, the online learning platform can analyze these data and feed them to the adaptive system. With these

advantages, online quizzes are prevalent in many education platforms, especially for math education, since students need plenty of practice to develop various math skills. Most online math quizzes would introduce diverse textual and visual content instead of abstract math expressions or text-only content to convey information and engage people by inspiring their interests [2].

1.2 Squla

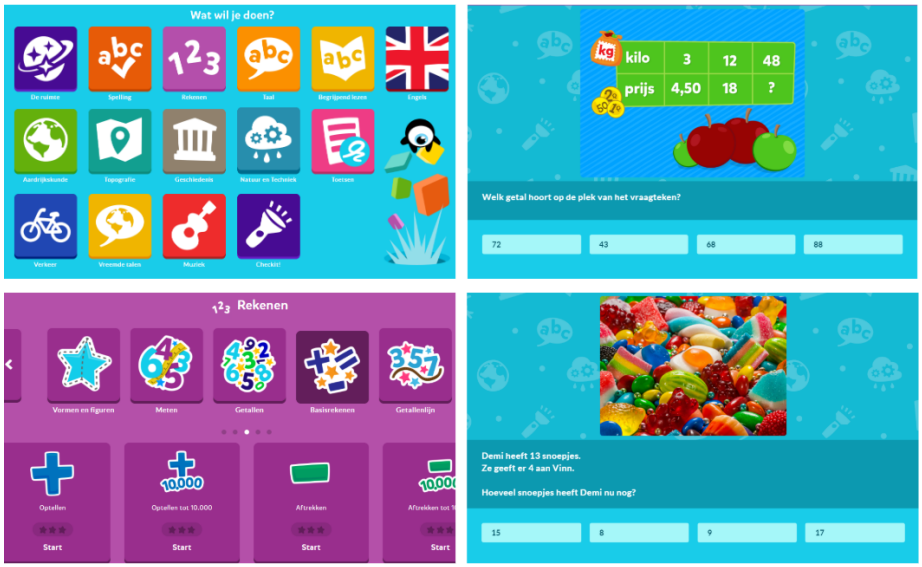


FIGURE 1.1: Interface of Squla webpage

This project is being done in collaboration with the company Squla. Squla is a company that has designed an interactive, fun educational online learning platform for primary school students in The Netherlands. Squla offers kids a convenient learning environment covering various subjects such as Math, Language, Spelling, History, Music, and Geography, in which children generally answer questions in quizzes to learn knowledge and improve their skills. For math education, Squla provides mathematical quizzes in various knowledge categories such as basic arithmetics, measure-

ments, word problems and comparison. In Squla, the math quizzes are designed and created by human experts based on hundreds of Knowledge Components defined by the national center of expertise for curriculum development (SLO). These knowledge components describe the specific math skills that are required for primary school students. Besides the underlying math problems, Squla also provides gamification elements, text stories, audio, and visual content to engage users' motivation.

1.3 Motivation

To offer students a convenient learning environment and to help them practice various mathematical skills, an online math learning platform needs a large amount and variety of content. For now, platforms such as Squla create content manually by human experts, which is extremely expensive and time-consuming. There is a large amount of repetitive, trivial work in content generation, especially for subjects like math. It is also inconvenient for editors to modify the content once created.

In contrast to producing content manually, Procedural Content Generation(PCG) aims at generating content automatically by algorithmic methods with limited or indirect human contributions. In addition to relieving human labor and time costs, another critical advantage for using PCG is that the automatically generated content can be updated in real time, which enables the algorithm to create non-repetitive diverse output. Squla has provided an adaptive system that can dynamically adjust the difficulty of the math quizzes. Quizzes in the database are assigned difficulty and confidence value within a specific range. With player modeling, these values are dynamically changed according to the users' previous performance when they answer the platform's quizzes. One characteristic of PCG is that the difficulty of generated math questions or puzzles can be controlled by adjusting the input parameters. Thus we are motivated to build a procedural content generator connected to such an adaptive system to keep the output aligned with adaptive difficulty levels.

Given these attractive benefits, PCG has been more and more popular in the past few decades. It has been successful in commercial games, like generating maps for dungeons in Spelunky and Zelda, generating personalized weapons in Spore, and generating terrains in Minecraft and Civilization IV.

Research shows that PCG would broadly accelerate the content generation process and increase the variety and novelty of the content [11]. The research on PCG receives an increasing amount of attention recently, but few concentrate on educational content generation, especially in mathematical education. Moreover, it is challenging to generate content covering a large variety of mathematical problems.

1.4 Research Questions and Main Contributions

Our project's goal is to explore PCG techniques to procedurally generate math quizzes with a variety of textual and visual appearance with limited and indirect human inputs, in order to support content editors and developers to accelerate the content generation process. This leads to the following research questions:

- How to keep the generated content aligned with age-specific curriculum and the desired difficulty?
- How can PCG accelerate the manual content generation process without compromising the content variety and quality?

To answer the first question, we analyzed and categorized the official curriculum-based knowledge components, extracting the quantitative requirements and constraints for generating math problems with specific difficulty levels. To answer both questions, we did an overall analysis of the content creation process in Squla learning platform and designed a procedural content generation pipeline that can accelerate this process. Meanwhile, we explored PCG techniques and methods for the textual and visual content generation to output various sensible math quizzes.

To the best of our knowledge, there are no methods proposed previously to procedurally generate textual and visual math education content aligned with curriculum-based knowledge and varying in difficulties and knowledge topics. Hence, our main contributions are:

1) We built a flexible, controllable, and designer-friendly content generator that can generate math problems in various categories.

2) We proposed a multi-language adaptive textual content generation approach to realize the abstract math problems into semantically coherent text questions in natural language.

3) We designed a rule-based generator of appropriate distractors for multiple-choice questions.

1.5 Thesis Structure

The thesis report is structured into seven chapters. The next chapter introduces the previous works that are related to our project, including PCG approaches for math education and natural language generation methods. Chapter 3 shows the overall generative approach, presenting an overview of the content generation pipeline. And then, we describe the abstract math problem generation method in Chapter 4. In Chapter 5, we describe the procedural generation method for text story and visual content. Chapter 6 presents and discusses the evaluation and results. Finally, we finish with the conclusion and outline ideas for future work in Chapter 7.

2

Related Work

This chapter first describes the related work and successful applications of procedural content generation in the field of math education. We discuss the availability and limitations of the previous research in generating various content for math learning platforms and educational games. In section 2.2, we concentrate on the related work on natural language generation methods available for textual math problem generation.

2.1 PCG approach for Math Education

So far, some procedural methods were introduced to generate various content for math education. One major research challenge is generating math problems at various difficulty levels. *MAST* is a web-based intelligent tutoring system for automated tutoring of probability problems [4]. The system allows users to select three parameters, including story problem header categories labeled with four challenge levels, problem contexts of coin or dice, and difficulty level of the statements of the questions. Each header category is corresponding to a rhetorical schema which encodes the related difficulty levels. The selected problem context and statement's difficulty

level specify the type of phrase and text spans used to realize the header schema into natural language text.

MentalMath [10] is an adventure game that requires the players to solve math problems as a final challenge after winning a shooting game at each level. They also employed a parametric-based method to generate text arithmetic problems based on templates. The parameters are specific to the templates' properties, such as the allowed maximum number's value, the number of digits in the corresponding arithmetic problem, and the fixed operator. Within the generated problems, they used a Pseudo-Random Number Generator (PRNG) to select a problem as the final challenge at the beginning of each level, and store it in an already used list.

Such parametric-based methods that encode the content properties make the generated problem controllable and maintainable. For generating problems in more varied difficulty levels, it is essential to find generic parameters that can control the difficulty quantitatively. However, in MentalMath, the difficulty of generated math problems is not specified. In MAST, the parameters all depend on numbered user's selection, which makes the generated content limited in flexibility and variety.

Polozov et al. [9] proposed an Answer Set Programming (ASP) based approach to generate arithmetic word problems from general user specifications, including tutor requirements and student requirements. This method produces personalized content, using declarative programming to search for solutions that satisfy the narrative requirements and constraints.

Refraction [12] is a flash puzzle game developed by Smith et al., aiming at education for understanding fractions through splitting and combining laser beams to form beams of fractional power. The levels in Refraction are generated automatically using two approaches separately: ASP-based approach and Depth-First-Search approach. Both of these approaches are used to generate the mission first, embed it into map grids, and finally generate the best solutions. The use of PCG with

logical programming techniques enables the content to be user-adaptive and with controllable difficulties. However, ASP-based methods used by Polozov et al. [9] and Smith et al [12] sometimes can be time-consuming due to their recursive nature, especially on non-search activities. The result achieves high accuracy and coherence but limited controllability and diversity of generated problem forms.

2.2 Natural Language Generation for math problems

Natural language generation (NLG) is the process of transforming structured data into natural, narrative language. This section describes the related work on NLG methods that focus on generating text stories as appearances of math problems, which can help students apply corresponding math skills under various contexts.

Typical conventional template-based methods use sentences or lists of phrases with open slots and insert words into these slots from predetermined word lists. MentalMath [10] employs a fully template-based method to generate text questions for arithmetic problems. What the generator does is replacing numbers in the manually designed sentence templates. The generated text problems are very repetitive in sentence structure and context.

Text generation from simple, highly structured templates is straightforward, quick, and easy to get accurate outputs. However, the generated texts are limited in variety, because the template format does not capture language analysis of different problem types and natural language complexities. Besides, the conventional template-based method is also clearly limited in maintainability, multilingual adaptivity, and output quality due to the random choice mechanic.

Khodeir et al. [4] realized the limitations of previous template-based text generation methods. To address those limitations, they exploited the rhetorical structure theory (RST) to control the text structure in their online tutoring system MAST. The RST-based methods perform well in generating complicated text structures by link-

ing minimal units such as short phrases recursively through their rhetorical relations. However, all the phrases are pre-defined manually, although the generated problems are various in logic structure, they are repetitive in vocabularies and statements.

Deane and Sheehan [3] proposed an automatic assessment item generation system based on Frame Semantics and a generic NLG pipeline. The pipeline consists of three modules: abstract document specification, logical schema generation, and sentence generation. First, they used an XML document to specify the possible values for the math problem variables and outline the sentence structure. Then the logical schema generator restructures the XML document into a series of frame semantics. Furthermore, the last step is to parse and wording the logical representations with grammatical information then output the natural language texts. The main difference between their NLG system and conventional template-based methods is that the abstract logic structure specification of text is separated from the details of sentence wording. The logical representation of text uses semantic frames that contain linguistic details, including the schematic structure and vocabulary patterns.

Polozov et al. [9] proposed a novel approach for math word problem generation. They divided the procedure into two steps: logic generation and natural language generation. With the input mathematical expression and the required constraints, a logical graph that consists of ontology types, relations, and discourse tropes can be generated using Answer Set Programming. After that, the system employs the same NLG pipeline as Deane and Sheehan [3] to generate the text question through natural language realization of the underlying logical graph,

The pipeline NLG architecture used by Deane [3] and Polozov [9] can be practical and generic for various tasks, contexts and multiple languages. However, both of them ignored the problem of how to phrase the sentence coherently and diversely. Thus, the generated texts are still limited in the variety of story plots. For students, especially primary school students, word problems with repetitive plots will make

them less engaged.

In order to address this problem and motivate students to be more engaged, Koncel-Kedziorski et al. [6] proposed a theme-rewriting approach for generating algebra word problems. Based on existing human-created word problems, the system rewrites those questions in new themes such as cartoons and Star Wars. The rewriting process contains two steps. First, the rewriting algorithm creates new texts by substituting thematically appropriate words and phrases. The new texts are then optimized using a measuring function quantifying syntactic, semantic, and thematic coherence. This approach uses no manual templates and achieves good results in generating non-repetitive, coherent, and solvable word problems.

Vijini et al. proposed a Long Short Term Memory (LSTM) model to generate textual math problems without any templates and predetermined structures [7]. With the big datasets of various content as input, the LSTM model would be trained to learn the likelihood of a word or character in a math word problem. The trained LSTM model takes a seed text with open slots as input and then generates the rest of the text. After that, the system uses POS (Part-of-Speech) tags to improve the accuracy of text by eliminating the problems of units, numerical constraints, and grammar mistakes. The deep neural network-based method overcomes the limitation of template-based and conventional NLG techniques on the creativity and novelty of the generated text. However, the approach has weaknesses in the accuracy of the generated text and the efficiency of training data.

2.3 Conclusion

Together, these studies indicate that procedural generation techniques can accelerate the creation of math content without or with limited human labor and generate content with good quality. Most of the applications and research focus on text question generation for arithmetic problems, and puzzle level generation for math

educational games that aim to practice specific knowledge skills. However, to offer a convenient learning environment for students, especially young children, we need a much larger variety of math problems, textual and visual contents. These relevant studies are valuable and inspiring for our project. We took advantage of the standard NLG architecture used by Polozov [9] and Koncel-Kedziorski's lexicalization method [6], and extended them to fit a generic pipeline that can generate various math problems.

3

General Approach

This project aims to design a procedural generation pipeline that can procedurally generate math quizzes for online learning platforms facing primary school students. Chapter 2 discusses various methods focusing on procedural math content generation and natural language generation. Still, most of these methods are limited in variety or quality of generated content and do not produce curriculum-based items.

To address this problem, we proposed a pipeline that divides the abstract math problem generation from the appearance generation, including textual and visual content. We proposed a template-based abstract math problem generation method to generate problems in various knowledge topics and combined techniques from natural language generation [9] and text scoring method [6] to realize the abstract problems into textual questions. Furthermore, we applied image retrieving and image generation to generate visual appearance.

Our generator's primary users are content editors, designers, and developers who are responsible for providing content in the online learning platform, aiming to support them to accelerate the process of content generation. The input is the required knowledge component selected by users and the language version of the text question.

The output will be a quiz that consists of the math problem corresponding to the selected knowledge component, the textual and visual appearance of the problem, the answer to the quiz, and distractors if the quiz type is a multiple-choice question. This Chapter describes the general approach (Figure 3.1) and its main phases: abstract math problem generation, distractor generation, and procedural appearance generation for the abstract math problems. The third phase is further divided into two steps: textual content generation and visual content generation.

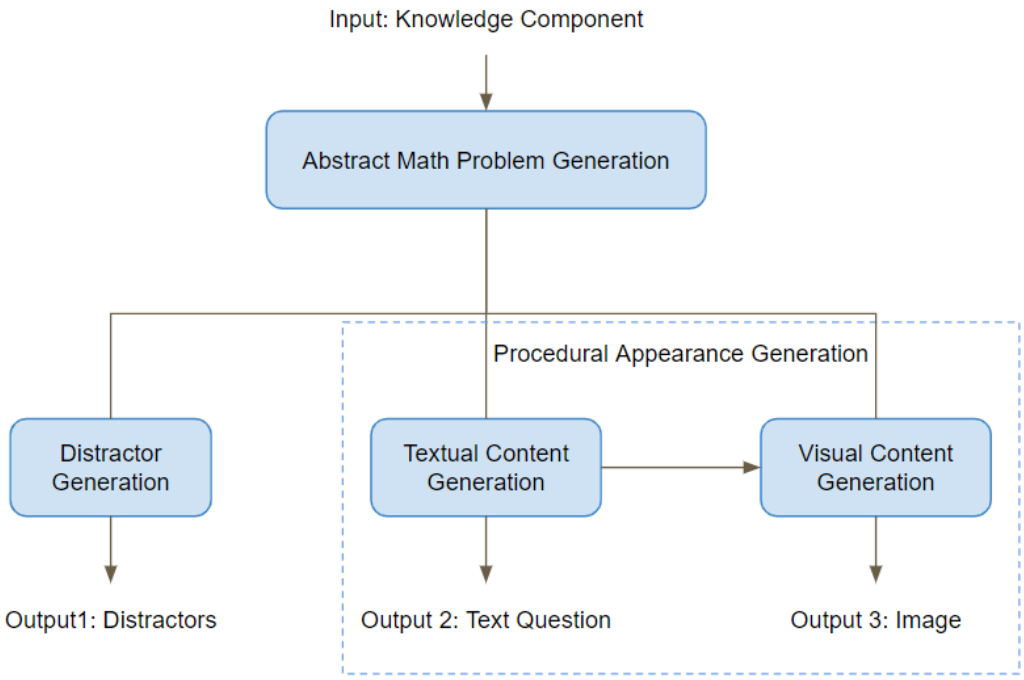


FIGURE 3.1: Pipeline of the General Approach

3.1 Abstract math problem generation

For primary school students, the math curriculum usually requires students to handle various math skills. In the Netherlands, as we mentioned in Section 1.2, the national center of expertise for curriculum development defines the required math

knowledge for different age-groups in the form of Knowledge Components. They contain the requirements, rules and constraints representing difficulty levels. As Table 3.1 shows, the Knowledge Components vary in math problem types including arithmetic, geometry, measurement, comparison, and mathematical relationships.

Table 3.1: Some knowledge components examples, the math problem category, and the abstract form.

Knowledge Component	Math problem category	Abstract form
Integer addition or subtraction within 100.	Arithmetic	Addition/Subtraction Equation
Equal sharing problems within 12.	Arithmetic	Division Equation
Compare integers within 20, and order them on a number line	Comparison	Sequence of integers
Compare the amount of money.	Comparison	Sequence of decimal numbers
Change length units from m to dm and cm, vice-versa.	Ratio Relationship	Ratio Table
Calculate new length and width values when resizing the images or maps.	Ratio Relationship	Ratio Table
Compare surfaces of flat objects.	Geometry	Flat Figures
Calculate the perimeter of a square or rectangle.	Geometry	Polygon

Each math problem category has one or several generic abstract forms. For example, in table 3.1, arithmetic problems can be abstractly presented by math expressions with operators and numbers. Ratio tables can abstractly present problems about ratio relationships. A sequence of numbers can abstractly represent varied comparison problems. Although one problem can be asked in different ways with varied textual and visual appearance, the basic underlying abstract form remains the same. Motivated by this, we designed the first phase of our pipeline to generate abstract math problems that satisfy the requirements of the input knowledge components. The term " abstract math problem" refers to the abstract form of the

math problem, which consists of mathematical expressions, symbols, tables, charts, or geometry information without any textual or visual appearance. We will describe the details of our method in the next Chapter.

Introducing the term "abstract math problem" enables our pipeline to generate the underlying math problems separated from the task of using textual or visual content to formalize the question. This makes the generator efficient to output a large number of problems with varied problem types and difficulty levels. However, this separation approach has cons at the same time. Except for the abstract problem, sometimes the structure of text questions can also influence the difficulty and the required math skills, which are not controllable in our generation pipeline.

3.2 Distractor generation

Multiple-choice questions are widely used to assess a student's knowledge of math concepts. They usually have four answer choices. Only one should be correct, and the other three should be distractors. Distractors in multiple-choice math questions should highlight mistakes that students commonly make. Manually designing these distractors is usually a time-consuming, repetitive work. In particular, much trivial calculation is needed for creating distractors for arithmetic problems. Common errors in these problems are generally based on a lack of understanding in operations such as computations, concepts, and procedures needed to be successful in a problem [5].

To explore methods for procedurally generating these distractors for arithmetic problems, we collected and analyzed some data of users' answers and managed to categorize the several most frequent mistakes they have made. Based on this, we designed three rules to generate the three distractors, respectively. Thus, this phase takes a generated abstract math problem as input and outputs the distractors for the corresponding multiple-choice question. We will describe the rule-based method in more detail in the next Chapter.

3.3 Procedural appearance generation

The last phase of the general approach aims to realize the generated abstract math problems into a concrete textual and visual representation. This phase consists of two parts: textual content generation and visual content generation.

Textual Content Generation The abstract problem generation phase has built a mathematical model of the problem. The first step of the procedural appearance generation phase is to generate natural language text questions for the underlying abstract math problem. Previous research has shown that the mathematical problem can be realized into text through two steps: logic generation and natural language generation. We introduced a combination of natural language processing techniques and the text scoring method to allow generation of semantically coherent, grammar correct text questions for various math problem types. Besides, this approach enables multiple language outputs with manually designed syntactic templates in different language versions. We will give a detailed explanation in Chapter 5.

Visual Content Generation Given the abstract math problem and generated text as inputs, the second step is to generate images related to the math problem. We adopted two methods for different purposes. One is generating images that consist of certain numbers of objects to convey quantitative information in the math problem (Section 5.2.1); The other is text-based image retrieval (Section 5.2.2) from an online website to present the input text question visually;

4

Procedural generation of abstract math problems

To answer the research question: *How to keep the generated content aligned with age-specific curriculum and the desired difficulty?*, we explored PCG techniques to generate math problems in various difficulty levels based on the curriculum requirements. Using controlled parameters is one of the most practical ways of generating math problems in different difficulty levels. Moreover, previous work shows that the template-based method offers a straightforward and effective way of generating math problems in the specified forms [10]. As discussed in Chapter 3, a variety of math problems can be categorized in several abstract forms. We can use templates defined by these forms to generate abstract math problems, with parameters to control the difficulty and other configurations required by the corresponding knowledge component. Thus, in this project, a template-based method was chosen to generate abstract math problems in various mathematical knowledge categories and various difficulty levels.

In this chapter, we describe the main steps of the abstract math problem generation approach, as shown in Figure 4.1. The input is knowledge components selected by users, and each knowledge component has its requirement and constraints, which

are manually extracted by us from its description. These requirements and constraints decide the difficulty and required knowledge skills of the math problems to be generated. We classified these knowledge components and built five generic templates that can generate abstract forms of math problems from the parameters that encode the requirements and constraints. In this phase, the input knowledge component is first classified to match a template to generate the abstract math problem. After that, the pipeline will automatically generate the answer and distractors using a rule-based method.

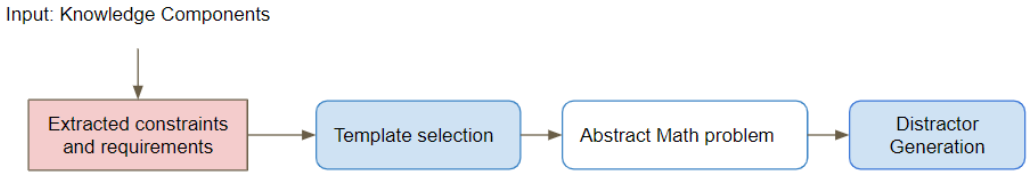


FIGURE 4.1: Main steps of generating abstract math problems and distractors

4.1 Content Classification

The national center of expertise for curriculum development(SLO) offers references for primary mathematical education, including over 600 knowledge components. The knowledge components define what math skills should be practiced and the quantitative constraints that define the required difficulty, such as the examples shown in Table 3.1. We analyzed and clustered similar characteristics among these knowledge components. The objective was to find a generic method to generate math problems that cover as many knowledge components as possible. Although some math problems are different in topics and knowledge categories, the abstract content follows the same formation. Finally, we classified 361 knowledge components into five groups and built templates for each group, to generate abstract math problems with the specific forms. The other knowledge components are excluded because they are

too complicated or unique to be classified, and we did not find the general forms for them.

4.2 Template Design

We built five generic templates for abstract math problem generation:

1. **Equation**

   Definition Mathematical expression with operands, operators on the left side of the equal sign, and a result number on the right side.

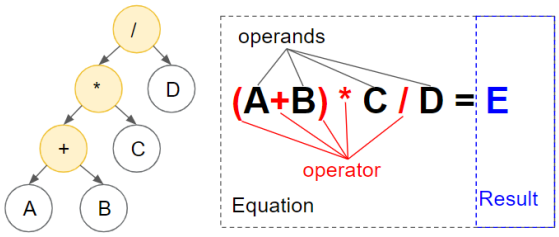


FIGURE 4.2: Equation structure

   Variables and parameters We extracted the requirements and constraints from the corresponding knowledge components and defined three parameters for equation structure:

   1) Operators.

   2) Range of operands.

   3) Range of the result.

As Figure 4.2 shows, these parameters ensure the system to generate an equation that satisfies the quantitative constraints of operands, operators, and results of the abstract arithmetic problem.

Method Taking the operands and operators generated from the constraint parameters, the equation is encoded as a binary expression tree to model the equation and calculate the result. Figure 4.2 shows the generic structure of an expression tree and the assignment of operands and operators to internal binary nodes.

2. **Number Sequence**

Definition A sequence of numbers in various data types including integer, decimal, fraction, time, date, temperature, and money value.

$$n_1, n_2, n_3, ..., n_i$$

Variables and parameters

We define four parameters in this template:

1) Value types.
2) Sequence length.
3) Range of Numbers, that limits the numbers within a specific range.
4) Specified Interval (optional), the default interval between numbers is random.

Function With different parameters given, the number sequence template can be used to generate different math problems. Given a specific interval, the generated number sequence can represent problems such as finding numerical

patterns and counting number lines. Given a random interval, it can generate numeric sorting and comparison problems. If the sequence length is one, the single number with a certain value type can be used to generate recognition problems such as recognizing temperature values. We present some examples of these problems in Appendix A.2

3. **Ratio Table**

Definition This template provides a table of proportional numbers that present the ratio relationship between the specific attributes.

Table 4.1: Ratio Table Template

Attribute 1	Attribute 2	Attribute 3	...
n_1	$r_1 * n_1$	$r_2 * r_1 * n_1$	...
n_2	$r_1 * n_2$	$r_2 * r_1 * n_2$	...
...	...	...	...

Variables and parameters Four parameters are defined here:

1) Ranges of the numbers in the first line. In Table 4.1, they are ranges of n_1 and n_2.
2) Ranges of Ratios that limit the ratios within several specific ranges. In Table 4.1, they are ranges of r_1 and r_2.
3) Attributes types. Some of them are encoded with semantic configurations such as "food" and "ingredient".
4) Number of rows.

Function This template can generate ratio tables that model different problem types with different attributes and number of rows and cols, such as unit conversion problem, distance-ratio-speed problem, etc. Examples are given in Appendix A.3

4. **Percentage Chart**

Definition This template generates totals and percentages as well as related charts or tables showing them.

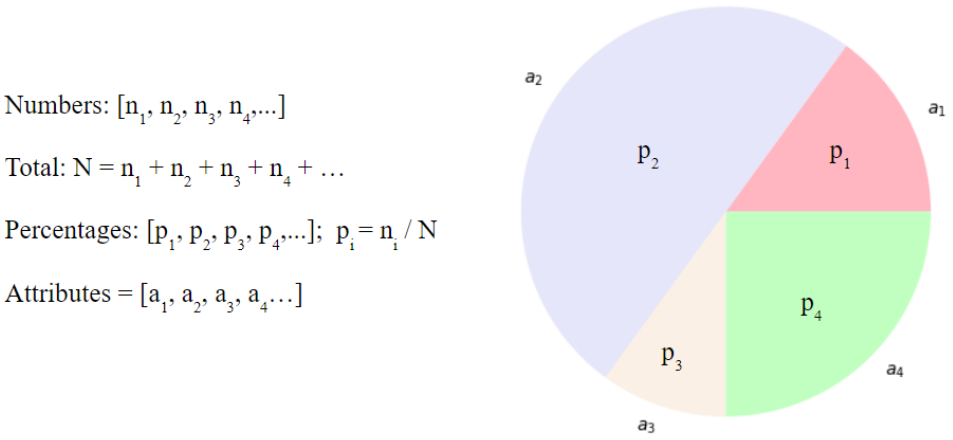


FIGURE 4.3: Percentage Chart Template

Variables and parameters Here we define four parameters:

1) Value type, including integer, fraction, decimal, and money value.
2) Chart type, including table, pie chart or bar chart.
3) Number of parts, that defines how many parts the total is divided into.

4) Range of percentages.

Function With the predefined attributes, this template can be used to generate various mathematical relationship problems. These generated abstract percentage charts enable students to practice skills for reading tables and charts, understand quantitative relationships between parts and whole, and solving percentage problems. We present some generated examples in Appendix A.4.

5. **Grid Paper**

Definition Basic geometry including triangle, rectangular, square, and combinatorial polygons, that are presented on a grid paper.

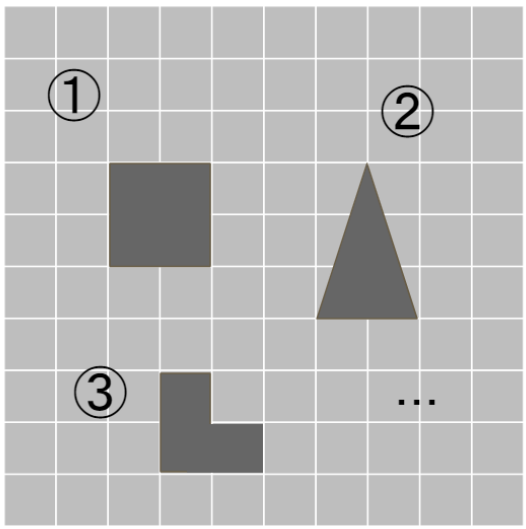


FIGURE 4.4: Grid Paper Template

Variables and parameters The parameters include geometry type (triangle, rectangular, circle, or polygon with a certain number of edges) and the number of geometry shapes. For polygons, we decide the geometry by the specified amount of vertices.

Function The grid paper helps children solve the geometry problem intuitively, such as measuring the area and perimeter by counting square boxes. When there is more than one geometry on the grid paper, it can represent the area or perimeter comparison problems. Some generated examples are presented in Appendix A.5.

4.3 Distractor Generation for Multiple-Choice Arithmetic Problems

Multiple choice question is a critical and popular way to test students' math skills. Our objective here is to define a series of rules to procedurally generate appropriate and distractive answers.

In our system, we realize the abstract problems generated from Template Equation, Ratio Table, and Percentage Chart into multiple-choice questions with four alternatives, including one correct answer and three distractors. For percentage problems, when the answers are numbers, the distractors are directly chosen from the generated percentage numbers. When the answers are charts, the distractors are similar charts with the same amount of parts. For problems generated from Template Number Sequence, the distractor generation is also straightforward by randomizing the sequence or choosing numbers from the sequence. However, for the arithmetic problem, the distractors are more tricky and indirect. Thus, this phase focus on generating distractors of multiple-choice questions for arithmetic problems, which are generated using Template Equation. To set reasonable and useful rules, we collected some user's answers in fill-in-blank arithmetic problems and found the three most

frequent types of wrong answers. They are:

1. Correct operation but wrong calculation.

   These wrong answers are caused by calculation errors, usually similar to the correct answer, with only one or two digits difference.

2. Correct calculation but using a wrong operation.

   Some students would use operators different from the given one, or not follow the correct order of the operation .

3. Wrong operation and wrong values.

   These wrong answers do not have a general pattern, but most of them are within a specific corresponding range, such as no more than two digits if the operands are both 1-digit numbers.

According to this, we design the following rules:

Rule 1. Generate a distractor similar to the correct answer:

Plus or minus 1 of the tail digit. For numbers that are multiples of 100 (e.g. 200 and 5000) and decimal numbers with more than one decimal digit (e.g. 0.01 and 5.125), generate the distractor by multiplying or dividing ten of the correct answer. Table 4.2 shows some generated examples using this rule in the third column.

Table 4.2: Examples: abstract math problems, the answers, and the generated distractors using the three rules, respectively.

Math Expression	correct answer	Distractor1	Distractor 2	Distractor 3
3 + 5	8	7	2	6
14 - 2	12	13	16	11
30 ×50	1500	150	80	1000
6 ×9	54	55	51	50
9 : 3	3	2	6	5
3 ×4 + 2	14	15	18	11

Rule 2. Switch operators or change the operation order:

Switch plus and subtraction, multiply to plus and division to subtraction. For some multiplication operation, we take advantage of multiplication rules, switching the operands during the operation process. For example, for 6×9, the correct result can be calculated by $6 \times 10 - 6$ generate the distractor by $6 \times 10 - 9$. For combined calculations, we use the incorrect order of operation and give the wrong result. The fourth column of table 4.2 shows some examples generated from this rule.

Rule 3. One random answer under certain constraints.

The distractor should have the same number of digits as the correct answer. Besides, for addition and multiplication, the distractor should be larger than the maximal operand. For subtraction and division, the distractor should be smaller than the minuend and dividend. The fifth column of table 4.2 shows the generated distractors from this rule. More examples of the generated dis-

tractors are given in Appendix A.1.

In addition to abstract math problems generated from Template Equation, Rule 1 and Rule 3 are also used in generating distractors for ratio problems generated from Template Ratio Table.

5

Procedural appearance generation for math problems

This chapter describes the pipeline of procedural appearance generation for the generated abstract math problems, as shown in Figure 5.1. To answer the research question:" *How can PCG accelerate the manual content generation process without compromising the content variety and quality?*", we designed a pipeline for textual and visual content generation, exploring the usage of natural language generation techniques in generating textual coherent and understandable text. We also used image generation and retrieving techniques to generate images related to the math question to make it more comprehensible.

This chapter gives a detailed explanation of the methods and techniques used to solve the problems outlined in the research question. Section 5.1 focuses on text generation for arithmetic, ratio, and percentage problems generated from the abstract math problem templates, describing how we realize these abstract content into natural language and explaining our chosen approach. Section 5.2 presents the visual content generation approach, and we specify it into two categories: image

generation and image retrieving.

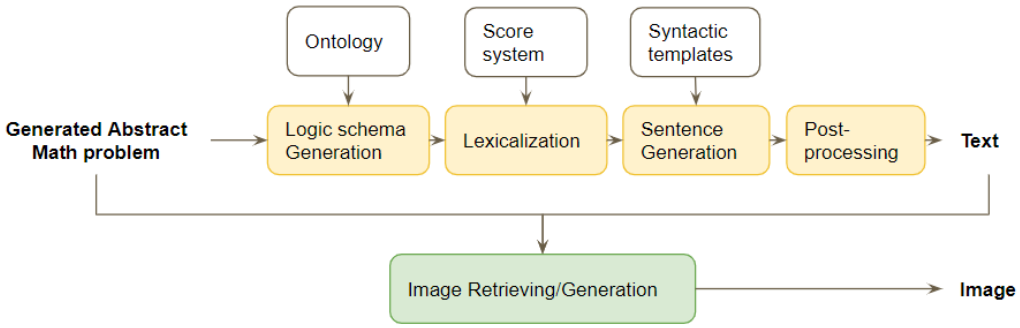


FIGURE 5.1: Main steps of procedural appearance generation for math problems

5.1 Text Generation

Justification: why design four steps in this phase, what's the advantage of generating logic schema first and using templates? According to the literature research, text generation for math problems generally consists of two parts: logic generation and natural language generation (NLG) [9, 3]. This general approach is adopted since separating these two parts enables us to design a controllable and flexible pipeline. Another alternative is to use manually created text questions as templates and replace several items such as numbers. However, this approach is very limited in variety and flexibility. The fully template-based method does not capture language analysis of different ontology types and does not consider natural language complexities. Thus, the advantage of our chosen method is that the abstract logic structure of the text is separated from the details of sentence wording, making the output not limited in the number of templates. Herein, we integrated Polozov, Koncel and Deane's methods [9, 3, 6]. In this project, we use the term "logic schema generation" to replace the generic term "logic generation". Logic schema generation refers to constructing a logic schema representation to encode the generated abstract math problem, which will be explained in detail in Section 5.1.1. Afterward, NLG tech-

niques are adopted to realize the logic schema into concrete textual representation.

This section presents the textual content generation pipeline used to realize the generated abstract math problems into text questions. As Figure 5.1 shows, the text generation module consists of four steps: logic schema generation based on the input abstract math problems and ontology relations and entities; lexicalization that turns the semantic configurations into coherent words using a scoring system; sentence generation using generic language-specific syntactic templates; and post-processing using POS-tagger to check grammar mistakes.

5.1.1 Logic schema generation

This module takes the generated abstract math problem as input, generating the logic schema of the underlying math problem with a predetermined ontology. A logic schema is structured by linguistic variables, which includes the following elements:

- Entities that are annotated with semantic labels that describe the patterns of vocabulary, such as "FOOD" and "VEHICLE" in Table 5.1.
- Ontology Relation word that defines the relationship between entities.

Table 5.1: Logic schemas of some example sentences.

No.	Logic schemas with Ontology Relations and Entities	Example Sentence
1.	HAS(OWNER,NUM=3,ITEM)	Tom has 3 apples.
2.	GIVES(GIVER, RECEIVER,NUM=2,ITEM)	Tom gives Jerry 2 apples.
3.	HAS(OWNER,NUM,ITEM); UNKNOWN(NUM)	How many apples does Tom have?
4.	BUY(SUBJECT, NUM=6,FOOD, LOCATION)	Anna buys 6 brownies from the cafe.
5.	COST(FOOD, NUM=2, EURO)	Each brownie costs 2 euros.
6.	SPEND(SUBJECT, NUM,EURO) UNKNOWN(NUM)	How much does Anna spend in total?
7.	DRIVE(VEHICLE, NUM=600, DISTANCE)	A car drives 600 miles.
8.	BE(FIGURE,AREA) UNKNOWN(AREA)	What is the area of the polygon?

Table 5.1 shows some examples of ontology relations and entities and the corresponding example sentences. The combination of the ontology relations and entities defines the logic of the text story that we are going to generate, and it is annotated with the mathematical problem. In the table, the first six ontologies can be used in the logic schema generation of the underlying arithmetic problems (3 + 2 = ?, 6 * 2 = ?), the seventh ontology is designed for ratio problem (distance-speed-ratio), and the last one is for geometry measurement problem.

For each abstract math problem category, one or more predetermined ontology types can be used for the logic schema generation of the text. The ontology defines the limitation of the vocabulary used in the sentence, which means there may be a limited number of word choices available for the substitution of variables in the logic schema. For example, as table 5.1 shows, for the variable "FOOD" in the logic schema, the word choices here should be in the category of food. This process will be described in detail in the next section.

As the research question suggests, we want to support the content editors and developers to accelerate the content generation process, without sacrificing the variety of the output content. Thus, we choose this approach instead of directly using manually created text questions as templates since, by using this approach, each ontology can be reused to generate various text stories, which enables much more diverse outputs than using the template-based approach. Besides, the logic schema's intuitive structure is more friendly to developers and content editors to modify or expand the list of ontology relations and entities to improve the system.

5.1.2 Lexicalization

The lexicalization module takes the logic schema as input, replacing the linguistic variables with concrete words. The linguistic variables in the logic schema are encoded with specific semantic labels, which should be substituted with words to construct a concrete sentence. For each semantic label, there is a word list that consists of the corresponding candidate words. For example, in the fourth logic schema in Table 5.1, the candidate words for the variable "FOOD" are: "brownie", "pizza", "cookie", "waffle", etc, and the candidate words for the variable "LOCATION" are "classroom", "playground", "cafe", etc.

To generate semantically coherent sentences, and increase the quality of the output text as the research question suggested, this module aims to choose the appropriate combination of words to generate reasonable and coherent text stories. We adopted a scoring method similar to Koncel's method [6] to evaluate the semantic coherence between the words retrieved from the candidate words list.

The lexicalization module takes logic schema L, and the linguistic variables $V = v_1, v_2, v_3...$ as the input. We represent the lexicalization output W in terms of the words that we choose to replace the variables list V, so $W = w_1, w_2, w_3....$ The ontology relation word is also included in W. First, we choose the words randomly

from the word list to construct a list of candidates W^*, and use a score function to find the best output among them. The score function quantitatively evaluates the semantic coherence among the words based on word embedding similarity provided in WordNet [8]. Formally, we define the scoring function S as:

$$S(W) = \sum_{i,j \in C} Similarity(\{w_i, w_j\})$$

where C is the set of word pairs from W, and $Similarity$ is defined as:

$$Similarity = cos(w_i, w_j)$$

where $cos(w_i, s_j)$ represents the cosine similarity between the embeddings of two words w_i and w_j.

Intuitively, if w_i and w_j are semantically coherence with each other, the similarity score would be relatively high, and vise versa. For the aforementioned example, ("brownie", "cafe") would gets a higher score than ("brownie", "playground"). After scoring the candidates, the system selects the word list with the highest score that is most semantically coherent. One advantage of this method is that it avoids giving unreasonable wording results, which improves the quality of the output text story. Another advantage is that it allows the usage of pre-trained word embeddings. Higher performance can be achieved with the embeddings that are trained from a larger text corpus.

5.1.3 Sentence Generation

As Figure 5.1 shows, the generated logic schema needs to be realized into a sentence in natural language. Thus, the sentence's underlying structure should depend on the constitution of ontology used in the logic schema. We designed a list of syntactic templates based on the ontology we used in the logic generation phase and matched these templates to the logic schema according to their attributes and the constitution

of ontology relations and entities. These templates are generic and primitive, which offer an effective way to generate sentences in correct word order and syntax.

This module realizes the lexicalized logic schema into a concrete sentence. The first step of this module is to select a syntactic template that matches the logic schema according to the entities' constitution and the ontology relation type. Afterward, the template produces a syntactic tree, an ordered, rooted tree representing the syntactic structure of the sentence according to some context-free grammar. Some subtrees, such as VP (verbal phrase) and NP (Noun Phrase) are replaced with the ontology relations and entities. As a result, a concrete sentence is obtained.

Figure 5.2 shows an example of a syntactic template that matches a logic schema L. These templates are language-specific. Here we use the English template as an example. If a different language version is needed, such as Dutch, we need to build new templates that have the correct grammar structure of the corresponding language.

After the system matches a pre-designed syntactic template with the input logic schema according to the ontology structure, the entities in the syntactic tree will be replaced by the words in the logic schema, and a rough, concrete sentence will be generated. As we can see in Figure 5.2, the generated text contains grammar mistakes related to issues such as plural mistakes for the nouns and missing third-person single transformation of the verbs.

5.1.4 Post Processing

To generate natural language with correct grammar and improve the quality of the text, we designed a post-processing mechanism using Natural Language Toolkit (NLTK) Part-of-speech (POS) tagging technique. The POS tagger can identify the subject, verb, numbers, units, objects, and adjectives in the sentence. Our system detects the tags and finds contradictions between them, such as the inconsistent tense

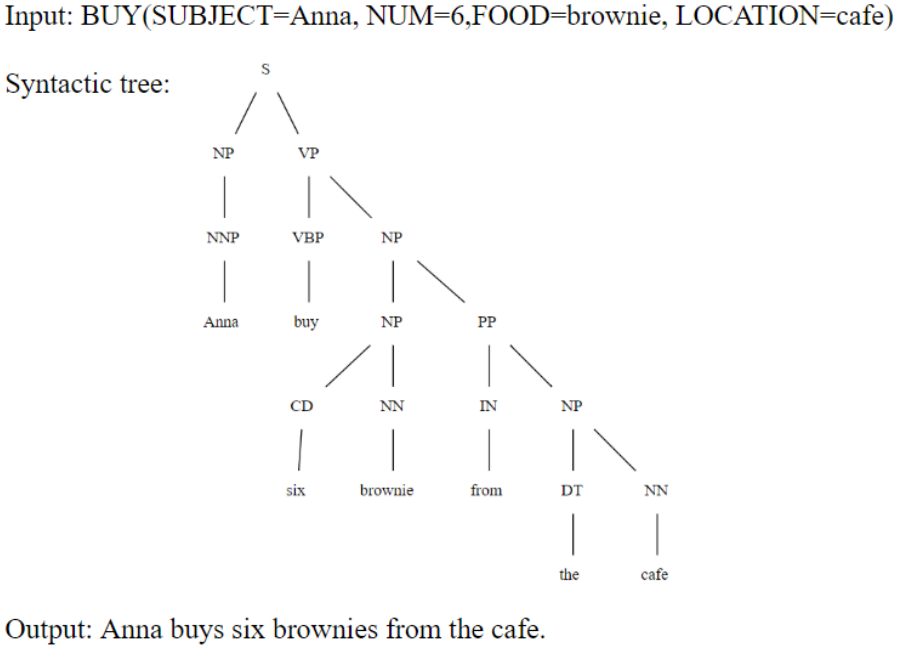


FIGURE 5.2: A syntactic template matches the input logic schema

of the verb and plural mistakes for the nouns. For example, after post-processing, the text in Figure 5.2 is turned into "Anna buys six brownies from the cafe.".

The POS tagging approach is adopted due to its practicality and effectiveness. The connection between numbers, objects, and subjects can be recognized to ensure the basic grammar correctness of the text. However, the output is not 100% accurate due to the nature of NLTK techniques. Therefore the generated text questions need to be checked or modified to make them 100% accurate.

5.2 Image generation and retrieving

Creating images and finding images online are time-consuming tasks for content editors. As the first research question suggests accelerating the content generation process, we explored the related methods and techniques for procedurally generating

the visual content after the text question is generated. According to our knowledge, students in lower grades of primary school are not required to solve the arithmetic problem with a math operation or using mathematical skills. Usually, they solve addition and subtraction problems by counting visible objects. Herein, we adopted an image generation approach to generate images with objects and the number of the objects generated is based on the numbers in the generated addition and subtraction problems. Note that the problems are restricted to only contain numbers less than or equals to 10. Moreover, after we generate the text question that provides a context for the math problem, a text related image is useful to enrich the quiz, making it more understandable and attractive to students. Therefore, our system generates related visual content in two ways: image generation for counting tasks and text-based image retrieving.

5.2.1 Image Generation

Here we focus on generating images with a certain number of objects. The purpose is to enable students to answer simple addition and subtraction problems within 10 by counting objects. First, we defined a list of objects such as apples and pencils that can be used in text questions for those arithmetic problems. Afterward, we made a list of images. Each image contains a single object in the middle with a white background. Given the number of objects specified by operands in the input equation, this module outputs a new image by splicing the object images together in a compact layout. The rules for the layout are:

- For subtraction problems, we generate the image containing objects equal to the minuend.
  - For even numbers, the objects are evenly distributed in the image.

  - For odd numbers, the objects are divided into two rows, and the difference between the number of objects in two rows is less than or equal to one. This layout is chosen because of its high readability and intuitiveness.

- For addition problems, we generate the image containing a plus sign in the middle, and the number of objects on the left side of the sign is equal to the first operand, and the number of objects on the right side of the sign is equal to the second operand. The layout of each side is as follows:
  - For numbers less than or equal to five, the objects are placed together in a row.
  - For numbers greater than five and less than ten, the objects are divided into two rows, and the difference between the number of objects in the two rows is less than or equal to one. Different from subtraction, a plus sign is presented in the image for the addition problem. Therefore we choose this layout to make the image compact and try to keep the length of objects on the left and right the same.

Figure 5.3 shows some generated images. More examples can be found in Appendix A.6.

5.2.2 Text-based image retrieving

For most math quizzes with textual context, text-related images are presented to attract users and to engage them in solving the problem. We adopted the text-based image retrieving method that retrieves the text-related images from the online image server. First, we extract the keywords in the text, such as objects, adjective words, and location adverbs recognized by the POS tagger in the post-processing step. Then the system takes these keywords as a search query and uses it to pull four most related pictures from the GettyImages API [1]. Herein, we only download

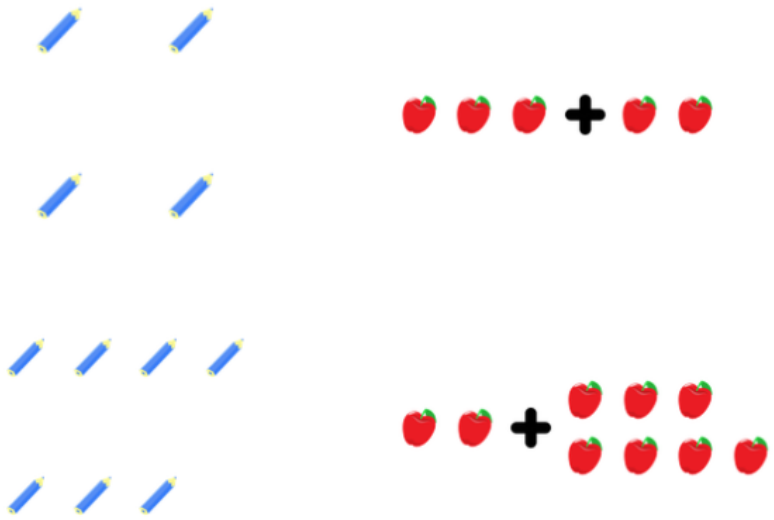


FIGURE 5.3: Four images generated based on the corresponding arithmetic operations. The left two images represent the operations with minuend 4 and 7 respectively. The right two images represent the operations "3 + 2" and "2 + 7".

the thumbnail-size form of these pictures. Each of them is annotated with a download link, which offers downloading access to the high-quality form. After retrieving the images, the content editors should check them and freely decide if they want to use the given download link to download one or more of the images or not. This approach is practical and fast, which offers an effective way to visualize the generated text. Content editors can decide to retrieve the images or not and select the images by themselves, which is more effective than manually searching images online. Figure 5.4 shows four retrieved images with the given text.

FIGURE 5.4: Four retrieved images using GettyImages API for the geneated text "Anna buys six brownies from the cafe.".

6

Evaluation

After we implemented the content generation pipeline, we had a quiz generator that generates text questions, answers, distractors, and images based on the input knowledge components. In this chapter, we evaluate these generated quizzes. These contents are evaluated in terms of quality and usefulness. To do this, we performed a human evaluation in which a sample of target users, including content editors and developers, were presented with a list of generated quizzes. The participants were asked to answer several questions which were designed around the research questions, and we will give a detailed explanation in Section 6.1.

6.1 Human evaluation

To understand the strengths and weaknesses of the generated content and to evaluate how helpful it is for our target users, we conducted a human evaluation participated by staff working in Squla. As discussed in the first chapter, our content generation pipeline's target users are content editors, designers, developers, and engineers responsible for providing content to the online learning platform. The participants of the evaluation questionnaire are these target users working in Squla who are re-

sponsible for content creation, designing, and engineers working about data analysis, player modeling, and platform development. Although the engineers do not design or create the content by themselves, they are knowledgeable and familiar with a large variety of content in the online platform, and provide technical support for the content editors. The participants were asked to score the generated math quizzes in three dimensions on the Likert scale from 1 to 5. The criteria are as follows:

1. Text Coherence

   This criterion is for evaluating our textual content generation method by asking how understandable and sensible is the text question.

   We sampled five generated arithmetic problems. Each of them has a text question that gives the context of the problem. The reason for choosing the arithmetic problems is that the contexts for these problems are not as straightforward as other problem categories such as order, measurement, and geometry. There is a story plot in each text question that helps students solve the mathematical problem in a practical context. Only if the story plot is presented with coherent text questions, the problem could be solvable and understandable to the students.

   Asking participants to score the text coherence is aiming to evaluate the quality of the generated text, which is suggested in the second research question. Generating coherent and sensible text story is also the main objective for us to design the textual content generation pipeline. Although the generated text can be artificial due to the nature of PCG and natural language generation techniques, the context in the textual question should be sensible, complete, and comprehensible for primary school students. Text questions that are not coherent would not be helpful and useful in the content generation process.

2. Quality of Distractors

   This criterion evaluates the distractor generation approach by asking how appealing and plausible the distractors are for primary school students.

   Appropriate distractors are important parts that constitute a good quality math quiz, which will significantly accelerate the manual content generation process and relieve the content editors' work of trivial calculation. We hypothesize these generated distractors from the generic rules that are plausible and reasonable for primary school students, although they can not be as adaptive to every specific arithmetic operation as manually designed distractors. We used the same five arithmetic problems as the problems used for text coherence evaluation. After the participants understand the context of the text question and the underlying mathematical problem, they are required to evaluate if the wrong answers are distractive and plausible.

3. Usefulness

   We evaluated the outputs' overall usefulness by asking how useful the generated content is for primary school students to practice the related mathematical skills.

   The content editors are supposed to design math quizzes with various textual and visual content that are useful to help students practice and improve their knowledge skills. Thus, evaluation of the usefulness of the procedurally generated content is an important step to decide whether content editors and designers can adopt them or not. This is also an overall evaluation of the quality of the generated math problems with textual and visual appearance. Besides, as the first research question suggests, the generated content should

be aligned with the curriculum and the desired difficulty. The quizzes beyond the curriculum requirements or with improper difficulty levels would not be useful for primary school students.

During the evaluation, if a low score was selected, the participants would be required to leave a short reason for the score, which could help us understand the weakness better.

6.2 Results

We collected the results from 17 participants who answered the questionnaire, including eight content editors, four platform developers, and five engineers working in data analysis and player modeling. Two of the content editors are educational experts in math. The questionnaire itself can be found in Appendix B . This section presents the overall results first, and then analyzes the results of the three criteria separately, discussing the evaluation findings and feedback.

6.2.1 Overall Result

To have an overall review of the evaluation results, herein we present and analyze the average scores of the questionnaire and the standard deviation. We are going to describe the results for each criterion in the next three sections in detail.

Table 6.1: Overall evaluation results

	Text Coherence	Quality of Distractors	Usefulness
Average	3.76	3.67	4.02
Standard Deviation	0.88	0.76	0.76

Table 6.1 shows the mean scores of the responses and the standard deviation. The first row shows the scores (with a minimum of 1 and a maximum of 5) of the corresponding questions that ask about text coherence, distractors' quality, and usefulness. As the table shows, the mean scores of all three criteria are greater than 3, which indicates that overall our pipeline produces useful content with good distractors and coherent text questions. It also indicates that the generated content can be leveraged by content editors and developers to provide understandable and solvable math quizzes. The standard deviations of the three criteria are less than 1, indicating that the scores are mostly centered on scores between 3 and 5, which means most participants are satisfied with the generated content.

6.2.2 Text Coherence

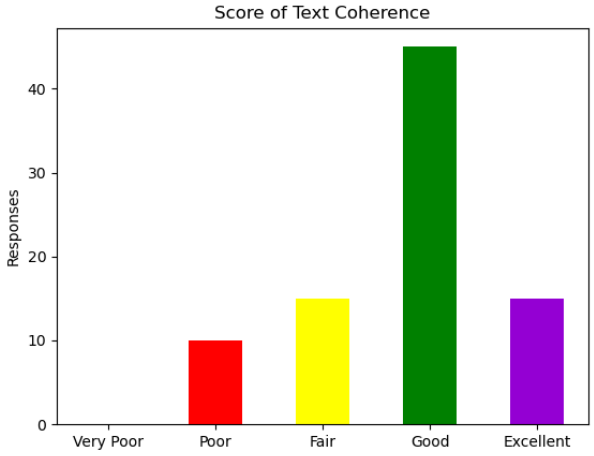


FIGURE 6.1: Bar chart showing the Text Coherence Score

We got 85 responses for text coherence evaluation. There is no "Very poor" response. 53% of the responses indicate the text questions are good in the text coherence evaluation, and 18% of them even represents that the text questions are excellent.

18% of the responses found the coherence of the generated text is fair. 11% of the responses chose the option "poor coherence", and they gave their reasons for the low scores. The main reason for the low scores is that the usage of words is not appropriate or friendly for young children, such as the usage of name "Fox", which may distract children to think about the animal fox. These wording issues can be avoided by manually revising the word library, eliminating the ambiguous and unfriendly words. Moreover, there is a feedback that the generated text looks artificial but still comprehensible.

6.2.3 Quality of Distractors

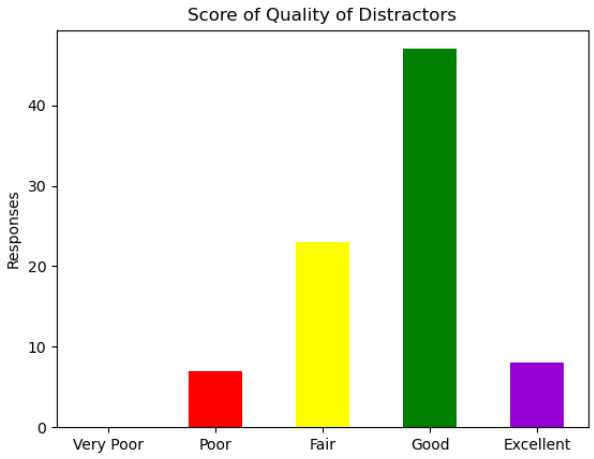


FIGURE 6.2: Bar chart showing the Distractors' Quality Score

We got 85 responses for evaluating the quality of distractors. 55% of them represent the generated distractors are appropriate and sensible, and 9% responses indicate the generated distractors are excellent. 27% responses rated the distractors fair. 8% chose the option "poor distractors". No one chose the "very poor" option. For low scores, their reasons are different from each other. The main reason is that

from different perspectives and with different habits, the participants found other distractors could make more sense. This indicates one of the weaknesses of procedurally distractor-generation methods is that the generated artificial distractors can not be perfect for all the math problems and are not as flexible as manually created distractors by educational experts.

6.2.4 Usefulness

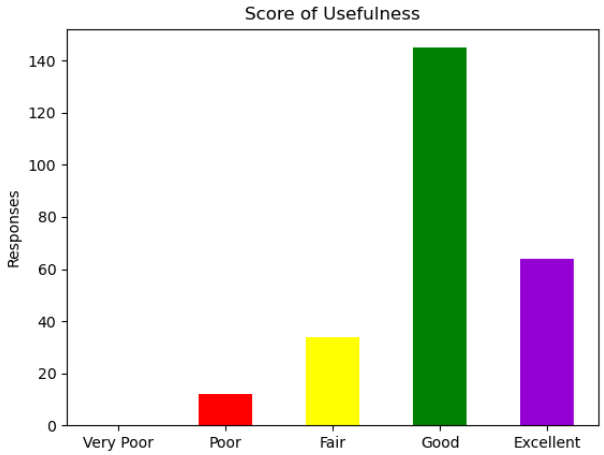


FIGURE 6.3: Bar chart showing the Usefulness Score

In addition to the quizzes that we used to evaluate text coherence and distractors, we sampled ten more quizzes generated from Template Number Line, Template Ration Table, Template Percentage Chart, and Template Grid Paper to evaluate the usefulness of generated curriculum-based content in various knowledge topics. Each quiz has a text question that describes the math problem and the related visual content such as a chart, table, or grid paper with figures. Herein we got 255 responses. 57% of the answers indicated that the generated quizzes are useful, with 25% chose the option "very Useful". One feedback is the generated tables could be a really nice

addition to the question set in the Squla database. 13% of the responses are "fair", and 5% are "poor". There is no "very poor" response. The reasons for low scores includes 1) the mathematical concepts such as "area" should be explained more; and 2) the context is not detailed enough. For example, in an arithmetic problem, the context is "Anna buys six hotdogs", without explaining why she would buy six hotdogs.

7

Conclusion

In this project, we explored the PCG techniques for math content generation and proposed a pipeline to procedurally generate math problems with textual and visual appearance. Our target users are content editors and designers, and our motivation is to help them accelerate the content generation process and support them in creating math quizzes in various knowledge topics and difficulty levels. Coming back to the research questions we raised in Introduction, the first question asks how to keep the generated content aligned with the curriculum requirements and the required difficulty levels, and the second question is about how can we use PCG techniques to accelerate the manual content generation process without compromising the content variety and quality. Our usage of the template-based method allows us to generate abstract math problems based on the knowledge components required in the primary school curriculum, with the difficulty level encoded in the knowledge components. These abstract problems are used as the input for the textual and visual appearance generation phase. Our design of separating the whole generation process into two phases enables the output to satisfy the curriculum requirements and makes it possible to generate questions in various knowledge topics. As for the second question,

the main idea of improving the content quality is to integrate natural language generation with optimized lexicalization techniques to generate coherent text questions with variable sentence structures and wordings.

We conducted a human evaluation to evaluate the procedurally generated content on the dimensions of text coherence, quality of distractors, and the overall usefulness. The evaluation result shows that these generated content, including curriculum-based math problems, distractors, coherent and sensible text questions, and the related visual content, are perceived to be of good quality and useful based on the evaluation result.

Currently, our procedural generation pipeline has two limitations. First, it is not possible to use templates to generate complex abstract math problems such as route planning on maps, or problems with complicated or special contexts. Second, the text generation module needs manually designed ontology structures and syntactic templates, which are hard-coded for now. In the future, it will be interesting to explore methods for automatically generating ontology for math problems to relieve these human efforts. Besides, we are willing to extend our content generation approach to other knowledge domains, for example, computer programming education for children by replacing mathematical concepts with declarative codes and visualizing the abstract code as game elements.

Appendix A

The Generated Questions

A.1 The Generated questions using Equation Template

A.1.1 English Version

Text Question	Abstract Problem	Answer	Distractors
Urga has 2 pencils. John gives Urga 8 pencils. How many pencils does Urga have now?	2 + 8 = ?	10	[100, 6, 15]
John has 5 cookies. John gives Julia 2 cookies. How many cookies does John have left?	5 - 2 = ?	3	[30, 7, 1]
Jedoo has 82 euros. Jedoo buys a coat for 15 euros. How much does Jedoo have left?	82 - 15 = ?	67	[77, 97, 63]
There are 2 black crayons and 75 white crayons in the warehouse. How many crayons are there in total?	75 + 2 = ?	77	[87, 73, 78]
There are 3 bags. Each bag contains 4 cookies. How many cookies are there in total?	4 * 3 = ?	12	[120, 7, 11]
Sam makes 3 pizzas. For each pizza Sam needs 3 nuts. How many nuts does Sam need in total?	3 * 3 = ?	9	[10, 6, 90]
Fox shares 6 apples with two friends. How many apples does each friend get?	6 : 3 = ?	2	[1, 3, 10]
Julia drives 72 kilometers in 9 hours. What is the average speed?	72 : 9 = ?	8	[80, 63, 5]
One pizza costs 6.7 euros. John buys 3 pizzas. One cake cost 2.8 euros. John buys 2 cakes. How much does John spend in total?	3*6.7+2*2.8 = ?	25.7	[25.9, 14.5, 28.7]

Text Question	Abstract Problem	Answer	Distractors
Er zijn 23 blauw shirten en 52 green shirt in the winkel . Hoeveel shirten zijn er in totaal?	23 + 52 = ?	75	[85, 29, 77]
Tom heeft 6 postzegels. Jedo heeft 18 postzegels. Hoeveel postzegels heeft Tom en Jedo in totaal?	18 + 6 = ?	24	[14, 12, 26]
Er zijn 22 snoepen in the zak. Tom neemt 15 snoepen. Hoeveel snoepen zijn er nog over?	22 - 15 = ?	7	[37, 4, 6]
Morgana heeft 5 bananen. Emma heeft 2 bananen. Hoeveel bananen heeft Morgana meer dan Emma?	5 - 2 = ?	3	[4, 7, 5]
Er zijn 7 zakken. elke zak bevat 9 potloden. Hoeveel potloden zijn er in totaal?	9 * 7 = ?	63	[53, 16, 59]
Morgana maken 7 taart. Voor een elke taart nodig hebben Morgana 8 bosbes. Hoeveel bosbes nodig hebben Morgana in totaal?	7 * 8 = ?	56	[560, 15, 55]
Anna deelt 21 postzegels met 2 vrienden. Hoeveel postzegels krijgen elke vriend?	21 : 3 = ?	7	[70, 18, 6]
Tom betalen 42 euro voor 7 taarten. Hoeveel kosten elke taart?	42 : 7 = ?	6	[5, 35, 9]
One taart kosten 1.0 euros. Zoe koopt 7 taarten. One perzik kosten 0.6 euros. Zoe koopt 10 perzikken. Hoeveel besteedt Zoe in total?	7*1.0 + 10*0.6 = ?	13	[23, 1, 10]

A.2 The Generated questions using Number Sequence Template

Question	Answer
Put the numbers in the correct order. From small to big. [7, 5, 6, 8, 4]	[4, 5, 6, 7, 8]
Choose the correct math sign: $<, > \; or \; =$ $1/4 \square 3/7$	$<$
Choose the correct math sign: $<, > \; or \; = \; .$ $8.45 \square 7.80$	$>$
Put the fraction numbers in the correct order. From small to big. [8/9, 1/7, 1/4, 4/7, 2/5]	[1/7, 1/4, 2/5, 4/7, 8/9]
Fill in the empty box. $[13, 14, 15, \square, 17]$	16
Fill in the empty box. $[31, \square, 37, 40, 43, 46, 49, 52]$	34
Put the numbers in the correct order. From small to big [489, 561, 681, 389, 891]	[389, 489, 561, 681, 891]
There are how many days between these two dates: 2020-01-04 and 2020-01-11?	7 days

A.3 The Generated questions using Ratio Table Template

- John bought 3 lemons for 2 euros. John needs 30 lemons. How much is it?

  Please filled in the empty box in the table.

number of lemon	price
3	2
30	?

- Please complete the equivalent fracture.

$$\frac{2}{7} = \frac{12}{?}$$

Answer: 42

Hint ratio table:

numerator	denominator	fraction
2	7	2/7
4	14	2/7
12	42	2/7
16	56	2/7
6	21	2/7

- Fill in the box:

$$82hm = \_\_\_\_\_km$$

Answer: 8.2

Hint ratio table:

km	hm	m
8.2	82	8200

- Fill in the box:

$$53dL = \_\_\_\_\_cL$$

Answer: 530

Hint ratio table:

L	dL	cL	ml
5.3	53	530	5300

A.4 The Generated questions using Percentage Chart Template

- The favorite color of the group is:

  14 choose blue

  10 choose green

  30 choose red

  21 choose yellow

  Which picture shows the correct distribution in the pie chart?

(a) (b) (c)

  Answer: (a)

- 50 children are asked what their favorite season is:

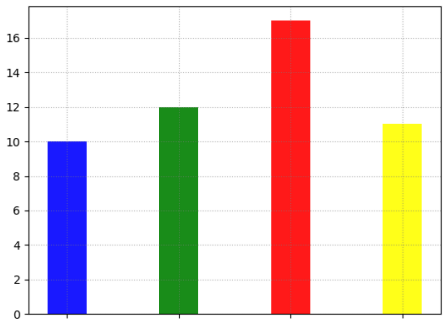
  10 choose Spring;

  12 choose Summer;

  17 choose Autumn;

  11 choose Winter.

  Look at the bar chart, which color of bar represents winter?

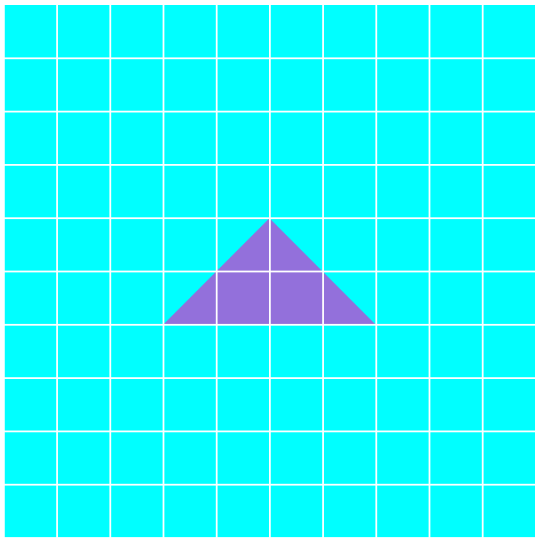


A. blue B. green C.red D.yellow

Answer: D

A.5 The Generated questions using Grid Paper Template

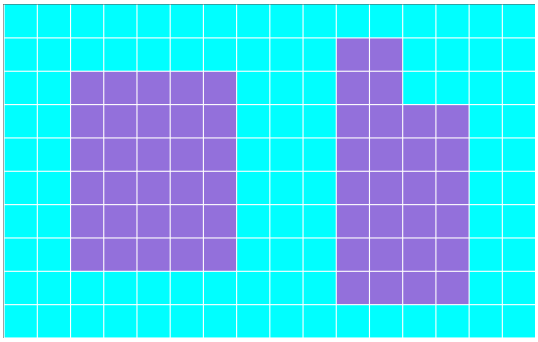
- Look at the picture. 1 box is 1 meter long and 1 meter wide. What is the area of the purple figure?



Answer: $4m^2$

- Question: Is the left polygon larger than the right one?

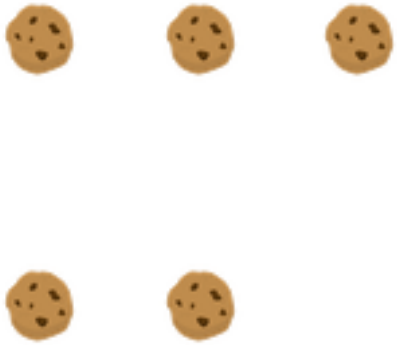
  A. yes B.no



Answer: A. Yes

A.6 Some questions with generated / retrieved images

- John has 5 cookies.John gives Julia 2 cookies.How many cookies does John have left?



A. 30 B. 7 C. 3 D. 1

- Anna has 2 apples. Brick gives Anna 3 apples. How many apples does Anna have now?

A. 1 B. 4 C.5 D. 50

- Zelph has 92 euros. Zelph buys a sweater for 88 euros. How much does Zelph have left?

A. 5 B. 4 C.180 D. 10

- Morgana has 19 cherries. Tom has 12 cherries. How many more cherries does Morgana have than Tom?

A. 7 B. 70 C. 31 D. 5

Appendix B

Questionnaire

Final evaluation form on procedurally generated math content

Dear colleague, this evaluation form asks you to assess the quality of automatically generated content in the field of math education. The questionnaire will show generated questions based on different knowledge topics, requesting you to score them.
*Required

1. Email address *

PART 1: Please evaluate the generated Multiple Choice textual questions for arithmetic problems.

We will show a generated text problem with multiple choice for several typical questions, requesting you to score them on three criteria:
1. Text coherence: Is the problem text understandable and sensible?
2. Quality of alternatives: are the distractors appealing and plausible to primary school students?
3. Usefulness: Is the generated content useful for primary school students to practice the related mathmetical skills?

2. There are 10 candies in the box. Fox takes 3 candies. How many candies are there left? A. 6 B. 13 C.7 D. 3 *

Mark only one oval per row.

	very poor	poor	fair	good	excellent
Text coherence	◯	◯	◯	◯	◯
Quality of alternatives	◯	◯	◯	◯	◯
Usefulness	◯	◯	◯	◯	◯

3. If you choose 'poor' or 'very poor' for any of the above questions, please give a brief explanation.

4. There are 16 butter cookies and 14 vanilla cookies in the gift box. How many cookies are there in total? A.2 B.20 C.30 D.36 *

Mark only one oval per row.

	very poor	poor	fair	good	excellent
Text coherence					
Quality of alternatives					
Usefulness					

5. If you choose 'poor' or 'very poor' for any of the above questions, please give a brief explanation.

6. One cake costs 7 euros. Emma bought 8 cakes. How much does Emma spend in total? A. 15€ B. 56€ C.66€ D.55€ *

Mark only one oval per row.

	very poor	poor	fair	good	excellent
Text coherence					
Quality of alternatives					
Usefulness					

7. If you choose 'poor' or 'very poor' for any of the above questions, please give a brief explanation.

8. Sophie has 8 candies. Sophie splits the candies to 4 boxes fairly. How many candies does each box have? A.4 B.5 C.6 D.2 *

Mark only one oval per row.

	very poor	poor	fair	good	excellent
Text coherence	◯	◯	◯	◯	◯
Quality of alternatives	◯	◯	◯	◯	◯
Usefulness	◯	◯	◯	◯	◯

9. If you choose 'poor' or 'very poor' for any of the above questions, please give a brief explanation.

_____

10. One sandwich costs 3,80 euros. One pizza costs 9,90 euros. Milan bought 7 sandwiches and 7 pizzas. How much does Milan spend in total? A. 91,90€ B, 95,90€ C.95.11€ D. 42.70€ *

Mark only one oval per row.

	very poor	poor	fair	good	excellent
Text coherence	◯	◯	◯	◯	◯
Quality of alternatives	◯	◯	◯	◯	◯
Usefulness	◯	◯	◯	◯	◯

11. If you choose 'poor' or 'very poor' for any of the above questions, please give a brief explanation.

_____

PART 2: Please evaluate the generated math problem for other knowledge topics/domains (e.g. number line, comparisons, ratio table, percentage, measurements, etc).

We request you to score them on the criteria "usefulness" - Is the generated content useful for primary school students to practice the related mathematical skills?

70

12. Question: Put the fraction numbers in the correct order. From small to big 1/9, 1/3, 2/9 Answer: [1/9, 2/9, 1/3] *

Mark only one oval.

- ◯ not useful at all
- ◯ useless
- ◯ fair
- ◯ useful
- ◯ very useful

13. If you choose 'useless' or 'not useful at all' for the above question, please give a brief explanation.

_____

14. Question: Fill in the empty box['16', '21', '_', '31'] Answer: 26 *

Mark only one oval.

- ◯ not useful at all
- ◯ useless
- ◯ fair
- ◯ useful
- ◯ very useful

15. If you choose 'useless' or 'not useful at all' for the above question, please give a brief explanation.

_____

16. Question: Choose the correct math sign: <, > or = 6.07 _ 6.13 Answer: < *

Mark only one oval.

- not useful at all
- useless
- fair
- useful
- very useful

17. If you choose 'useless' or 'not useful at all' for the above question, please give a brief explanation.

_____

18. Question: Fill in the box: 7500m = _km Answer: 7.5 *

Mark only one oval.

- not useful at all
- useless
- fair
- useful
- very useful

19. If you choose 'useless' or 'not useful at all' for the above question, please give a brief explanation.

_____

20. Question: John bought 3 hotdogs for 6,70 euros. John needs 30 hotdogs. .How much is it? Please filled in the question mark. *

number of hotdog	price(euro)
3	6,70
30	?

Mark only one oval.

- ◯ not useful at all
- ◯ useless
- ◯ fair
- ◯ useful
- ◯ very useful

21. If you choose 'useless' or 'not useful at all' for the above question, please give a brief explanation.

_____

22. Zoe drives 528 kilometers in 6 hours. What is the average speed? __km/h
Answer: 88 *

Mark only one oval.

- ◯ not useful at all
- ◯ useless
- ◯ fair
- ◯ useful
- ◯ very useful

23. If you choose 'useless' or 'not useful at all' for the above question, please give a brief explanation.

_____

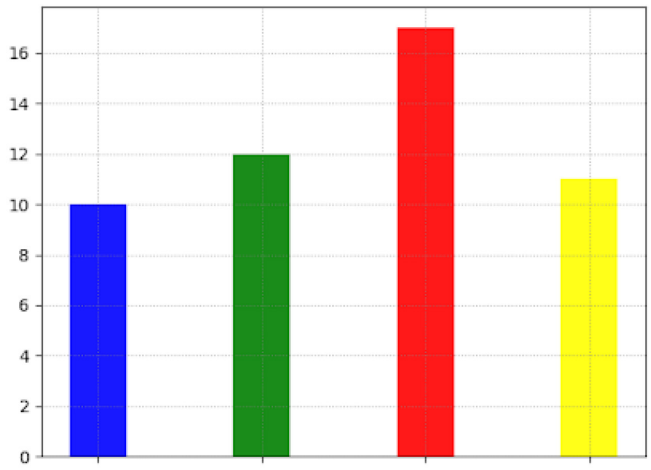
24. The favorite color of the group is: 30.0% blue 50.0% green 20.0% red Which picture shows the correct distribution in the pie chart? *

Mark only one oval.

- ◯ not useful at all
- ◯ useless
- ◯ fair
- ◯ useful
- ◯ very useful

25. If you choose 'useless' or 'not useful at all' for the above question, please give a brief explanation.

26. Question: 50 children are asked what their favorite season is: 10 choose Spring; 12 choose Summer; 17 choose Autumn ; 11 choose Winter. Look at the bar chart, which color of bar represents winter? A. blue B. green C.red D.yellow *

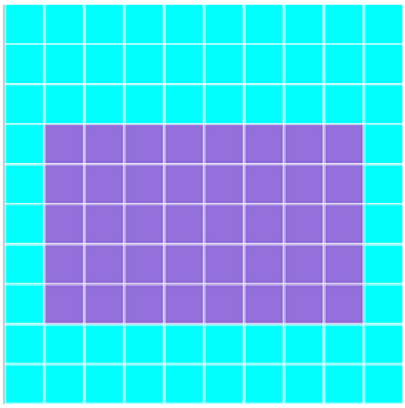


Mark only one oval.

- not useful at all
- useless
- fair
- useful
- very useful

27. If you choose 'useless' or 'not useful at all' for the above question, please give a brief explanation.

28. Question: Look at the picture. 1 box is 1 meter long and 1 meter wide. What is the area of the purple figure? __m² Answer: 40 *



Mark only one oval.

- not useful at all
- useless
- fair
- useful
- very useful

29. If you choose 'useless' or 'not useful at all' for the above question, please give a brief explanation.

Bibliography

[1] `https://developers.gettyimages.com/en/`.

[2] Simon KS Cheung, Will WK Ma, Lap-Kei Lee, Harrison Yang, et al. *Blended Learning. New Challenges and Innovative Practices: 10th International Conference, ICBL 2017, Hong Kong, China, June 27-29, 2017, Proceedings*, volume 10309. Springer, 2017.

[3] Paul Deane and Kathleen Sheehan. Automatic item generation via frame semantics: Natural language generation of math word problems. 2003.

[4] Nabila Ahmed Khodeir, Hanan Elazhary, and Nayer Wanas. Generating story problems via controlled parameters in a web-based intelligent tutoring system. *The International Journal of Information and Learning Technology*, 2018.

[5] Kelly V King, Doug A Gardner, Sasha Zucker, and Margaret A Jorgensen. The distractor rationale taxonomy: Enhancing multiple-choice items in reading and mathematics. *Assessment Report. Pearson*, 2004.

[6] Rik Koncel-Kedziorski, Ioannis Konstas, Luke Zettlemoyer, and Hannaneh Hajishirzi. A theme-rewriting approach for generating algebra word problems. *arXiv preprint arXiv:1610.06210*, 2016.

[7] Vijini Liyanage and Surangika Ranathunga. A multi-language platform for generating algebraic mathematical word problems.

[8] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[9] Oleksandr Polozov, Eleanor O'Rourke, Adam M Smith, Luke Zettlemoyer, Sumit Gulwani, and Zoran Popović. Personalized mathematical word problem generation. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[10] Luiz Rodrigues, Robson Parmezan Bonidia, and Jacques Duílio Brancher. A math educacional computer game using procedural content generation. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 28, page 756, 2017.

[11] Noor Shaker, Julian Togelius, and Mark J Nelson. *Procedural content generation in games*. Springer, 2016.

[12] Adam M Smith, Erik Andersen, Michael Mateas, and Zoran Popović. A case study of expressively constrainable level design automation tools for a puzzle game. In *Proceedings of the International Conference on the Foundations of Digital Games*, pages 156–163. ACM, 2012.