Procedural generation of problems for elementary math education

Yi Xu[1], Roger Smeets[2], and Rafael Bidarra[1]
[1]*Delft University of Technology, The Netherlands*
[2]*Futurewhiz, Amsterdam, The Netherlands*

Abstract

Mathematics education plays an essential role in children's development, and there are many online applications aimed at supporting this process. However, manually creating math problems with a variety of textual and visual content is very time-consuming and expensive. This article presents a generic approach for procedural generation of mathematical problems, including their corresponding textual representations. The content generation process consists of two phases: abstract math problem generation and text generation. For the generation of abstract math problems, we propose a generic template-based method that operates across a variety of difficulty-levels and domains, including arithmetic, comparison, ordering, mathematical relationships, measurement, and geometry. Subsequently, we propose a multi-language adaptive textual content generation pipeline to realize the generated abstract math problems into semantically coherent text questions in natural language. A workflow time gain evaluation shows that the system yields an average time saving of 56%. Further, human expert evaluation of this approach indicates that the content it generates is sensible and solvable for primary school students.

Keywords: Procedural content generation; math problem generation; mathematics education; online education.

1 Introduction

Online education has become more prevalent at virtually all levels of education – a trend which has been reinforced during the onset of the COVID-19 pandemic. Digital educational services are offered in a wide variety of flavors, such as Massive Open Online Courses (MOOCs), intelligent tutors, or educational games.

One provider of so-called "edtech" services is Squla, an Amsterdam-based company that offers educational games for children aged 4-12 in the Netherlands and Poland. At the time of writing, it has a collection of 122,138 distinct questions, spread across 38 different subjects and eight grades. Questions can be of several types – e.g. multiple choice, open answer, or fill-in-the-blanks – and are offered in various ways to users – e.g. via answer selection, drag-and-drop, or catapult shooting. Unsurprisingly, creating such new content is a laborious, time-consuming and hence expensive process. Among other tasks, it involves abstract problem formulation, context/story generation, selection or creation of visuals, and quality control. Typically, two to three people are involved in this process, usually content editors and designers, and the cost of creating one new question can easily run in the hundreds of euros.

These challenges of content creation become particularly salient when one also aims at providing adaptive gameplay [1, 2]. In this setting, question difficulty should automatically adapt to the abilities and progress of the player [3]. In order for the underlying algorithms to function properly, a large and dense corpus of questions at varying levels of difficulty has to be carefully indexed, further increasing the burden of content creation.

In contrast to manual content production, Procedural Content Generation (PCG) aims at generating content automatically by algorithmic methods, with limited or indirect human intervention. Research shows that PCG can significantly accelerate the content generation process, and even increase its variety and novelty [4]. In this context, this research aims at *exploring how PCG techniques can support human content editors (like those at Squla) in the generation of math questions for elementary education*.

A number of specific challenges have to be addressed in this regard. First, and foremost, the human content creators should have adequate degree of control over the content generator (e.g. in terms of themes, objects and images used). Second, the PCG process should generate textual content without compromising quality and variety. Third, the PCG process should align with the dynamic requirements of content topics and difficulty, which may also depend on the learner profile (e.g. age, skills, etc.).

We have designed a procedural generation pipeline for math content that can generate math problems with textual and visual representation, while taking only one input: a so-called "knowledge component", which captures the requirements in terms of both topic and difficulty level. Our approach splits the content generation process into two phases: abstract math problem generation and text generation. We propose a generic template-based method to generate abstract problems in a variety of math topics. Moreover, we developed a multi-language adaptive textual content generation pipeline to translate the abstract math problems into semantically coherent text questions in natural language.

2 Related work

PCG in math education PCG is the algorithmic creation of digital content with limited or indirect user input, and it is being increasingly applied in the design of games, puzzles and virtual environments [4–6]. Since at least two decades ago, PCG has been proposed to stimulate students for learning math. Santos et al. [7] proposed the Computer-Aided Learning (CAL) system, which allows designers to write template multiple-choice questions, and generates different instances of these templates for the assessment of Linear Algebra courses. The authors exemplify this template-based method for generating matrix problems based on mathematical constraints. They also point out that it is an easy task for programmers, but not straightforward for teachers and content designers.

In recent years, PCG has been gaining increasing attention for educational purposes and some novel procedural methods were introduced to generate content for math education. Smith et al. [8] put forward that PCG is not only useful for improving the replayability and adaptability of the game content, but also valuable for fostering collaborative mindful learning. Players are likely to share and discuss the game content among a community, and the large variety of the procedurally generated content allows the players to discuss the reasons behind the answers rather than the answers themselves. 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. Refraction, a flash puzzle game developed by Smith et al. [10], also employs ASP to specify the content constraints by searching the best solutions to generate missions and solvable puzzles. Such ASP-based method uses declarative programming to search for solutions that satisfy the input require-

ments and constraints. This method performs well in difficult combinatorial search problems but sometimes it can be time-consuming due to its recursive nature, especially on non-search activities.

Hooshyar et al. [11] proposed a data-driven PCG approach for educational games using support vector machines to build the fitness function for a genetic algorithm, which provides adaptive game content tailored to students' proficiency. Singhal et al. [12] proposed an automated geometry question generation framework for high school students, which allows users to select geometric objects, concepts and theorems. They used a deductive approach that first generates geometric figures with a set of unknown variables based on the users selection, and after that it generates the facts and solutions to find values of the unknown variables which represent the relationships between the geometric objects.

One major research challenge of PCG for math education is generating math problems at various difficulty levels. Rodrigues et al. introduced MentalMath [13], which is an adventure game that requires the players to solve math problems as a final challenge after winning a shooting game at each level. The authors employed a parametric-based method to generate text arithmetic problems based on templates. The parameters are specific to the templates' properties, such as the maximum values allowed, the number of digits in the corresponding arithmetic problem, or the operators allowed. Khodeir et al. designed a web-based intelligent tutoring system *MAST* for automated tutoring of probability problems [14]. The system can be configured by the user to control the problem context and statement's difficulty level. Such parametric-based methods that encode the content properties make the generated problem controllable and maintainable. However, due to very specific and limited parameters, the generated content is restricted in flexibility and variety. For generating problems in a wide range of difficulty levels, it is essential to find generic parameters that can better control the content difficulty.

Natural Language Generation for Math Problems Natural language generation (NLG) refers to the process of transforming structured data into natural, narrative language. A number of projects have explored NLG techniques for generating text stories as realizations of math problems. MentalMath [13] employs a fully template-based method to generate text questions for arithmetic problems by replacing numbers in the manually designed sentence templates. Text generation from such simple, highly structured templates is straightforward, quick, and easy to get accurate outputs. However, the generated texts are very limited in variety.

Khodeir et al. [14] realized the limitations of previous template-based text generation methods and employed rhetorical structure theory (RST) to generate the problem context in their online tutoring system MAST. The user's selection specifies the type of phrase and text spans used to realize the header schema into natural language text. The RST-based methods perform well in generating complicated text structures by linking minimal units such as short phrases recursively through their rhetorical relations. However, all the phrases are manually pre-defined and, although the generated problems are varied in logic structure, they are repetitive in vocabulary and statements.

Deane and Sheehan [15] proposed an automatic assessment item generation system based on Frame Semantics and a generic NLG pipeline. 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. The last step is parsing and wording the logical representations with grammatical information, and output the natural language texts. Polozov et al. [9] employed an NLG pipeline similar to Deane and Sheehan, to generate the text question through natural language realization of the logical graph generated based on the required math expressions. Both Polozov and Deane's

methods are dependent on predefined word phrases to formalize the text problem. Wyse et al. [16] proposed a rule-based approach based on pre-defined question templates. The system takes syntax trees as input and uses specific rules for pattern matching to generate questions based on the matched templates.

Gupta et al. built a tool called Intelligent Math Tutor (IMT) [17], which procedurally generates math word problems embedded with teachings from other subjects and concepts, to help students learn knowledge in multiple domains in parallel. They used a conceptual entity extraction method to highlight the entities in math word problems and the relationships between them. Using the generated concept map, the system matches the math problem with concepts in other knowledge domain that have a similar map structure, and blends the two pieces of text to generate a new math problem. This novel idea and approach were successful in generating interesting and meaningful math problems. However, the aforementioned methods are limited in output variety as they did not take into account the linguistic properties other than the syntactic functions and sentence structures.

Koncel-Kedziorski et al. [18] proposed a theme-rewriting approach for generating algebra text problems consisting of two steps. First, the rewriting algorithm creates new texts by substituting thematically appropriate words and phrases. The new texts are then optimized using a metric function that quantifies syntactic, semantic, and thematic coherence. This approach uses no manual templates nor predefined word phrases and achieves good quality and variety in generating variants of existing word problems. Their method uses Word2vec [19], a popular approach to learn word associations from a large corpus of text, representing words on a fixed-length vector space model [20]. Koncel's theme-rewriting approach selects coherent text stories with appropriate word phrasing by using cosine similarity to compute the semantic similarities between the word embeddings.

Nandh ini et al. [21] compared two popular math text problem generation approaches: *template-based NLG*, which generates variants of the template questions by replacing the semantic representations without changes on the syntactic structure; and *grammar-based question generation*, which preserves the semantics of the questions and generates questions in different syntax patterns. Our method partly integrates those two approaches, aiming at getting the best of both (semantic and syntactic) worlds, while providing control over the increased variety of the generated text questions.

Liyanage et al. proposed a Long Short-Term Memory (LSTM) model to generate textual math problems without any templates and predetermined structures [22]. 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 issues with units, numerical constraints, and simple grammar mistakes. This 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 accuracy of the generated text and the efficiency of the approach is very dependent on the quality of the training data. In addition, once trained, the generation method is not really controllable nor expansible by a human content creator.

Summarizing, current methods for math problem generation which promise a reasonable semantic correctness have limitations in their output diversity and/or in the amount of control they provide to steer that variety. Conversely, when a large variety is supported, it often suffers from weak semantic or narrative coherence, providing a somewhat 'flat experience'. Moreover, when some control over the output generation is provided, it is mostly too specific, narrow or requiring a technical background, which most content creators lack. While these drawbacks may be acceptable for many applications (e.g. an unsupervised online quiz, or a casual captcha), a more reliable solution is required when high-quality educational content

has to be massively deployed online to many thousands of students. Our proposal, as outlined in the next section, provides a solution to the above dilemma.

3 General approach

The SLO, the Dutch national center of expertise for curriculum development, provides a framework for primary mathematical education, including the description of hundreds of learning objectives. These learning objectives – henceforth referred to as *knowledge components* – constitute a very convenient and generic notion representing both a math topic and its associated constraints.

In order to address the generation of curriculum-based math exercises in a broad difficulty range and with varied textual and visual realizations, we propose a generic content generation pipeline that takes a desired knowledge component (KC) as input. The output is a quiz consisting of (i) an abstract math problem corresponding to that KC, (ii) the textual and visual realization of the problem, (iii) the correct answer and, possibly, (iv) so-called distractors (if the quiz type is multiple-choice).

Figure 1 shows the architecture of our general approach. 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; it is thus prior to creating any textual realization or visual appearance.

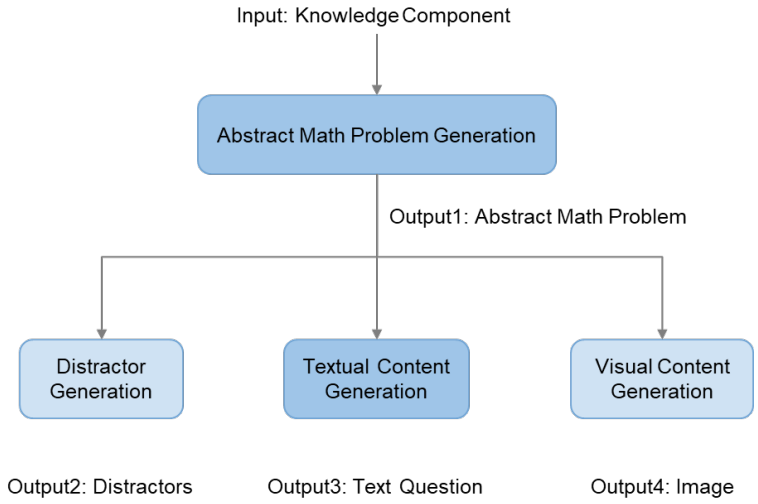

Figure 1: *Pipeline of the General Approach*

Our approach handles the generation of the underlying abstract math problems separately from the task of choosing textual or visual content to realize the question. This enables the generator to efficiently generate a large number of problems with controllable difficulty levels and desired problem types. To generate abstract math problems, we propose a template-based method using five generic, parameterized abstract templates. The abstract templates' parameters and variables enable us to control the difficulty and the properties of math problems. These properties of the underlying math problems typically bear no relationship with the context, text question, or images. For example, the same abstract arithmetic problem $5+12+17=?$ can be realized referring to minutes, Euros or candies.

The text generation phase takes the abstract problem as input, and realizes it into a natural language text question. We use a combination of NLG techniques inspired on those of Deane [15] and Poloznov [9], as well as the text scoring method of Koncel-Kedziorski [18] to

effectively support the generation of semantically coherent, grammatically correct text questions for various math problem types. To handle the answer generation for multiple-choice questions, we propose a rule-based distractor generation approach to generate distractive and plausible wrong answers based on the given abstract problem. Moreover, we also developed a simple visual content module that automatically retrieves from a large database appropriate images thematically related to the text realization of the generated abstract problem. In the next two sections, we describe in more detail the two main phases of this approach (dark-shaded boxes in Figure 1).

4 *Procedural generation of abstract math problems*

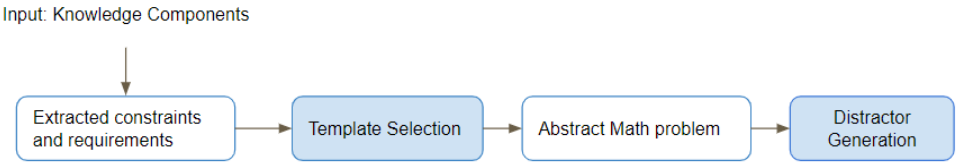

Figure 2: *Main steps of generating abstract math problems and distractors*

The main steps of the abstract math problem generation approach are shown in Figure 2. The shaded boxes represent the main steps of this phase. The input consists of knowledge components, each of which has its topic (e.g. arithmetic), requirements (e.g. addition) and constraints (e.g. within 100). An input KC is first classified in order to select an appropriate template for the abstract math problem generation. Afterwards, the pipeline will automatically generate the answer and, possibly, the distractors.

Table 1: *Sample of the database of knowledge components, their math problem category, and the corresponding abstract form*

Knowledge Component	**Math problem category**	**Abstract form**
Integer addition or subtraction within 100	Arithmetic	Equation (Addition/Subtraction)
Equal sharing problems within 12	Arithmetic	Equation (Division)
Compare integers within 20, and order them on a number line	Comparison	Number Sequence (Integers)
Compare the amount of money	Comparison	Number Sequence (Decimal numbers)
Change length units from m to dm and cm, vice-versa	Mathematical Relationship	Ratio Table
Read the percentages from a pie chart	Mathematical Relationship	Percentage Chart
Compare surfaces of flat objects	Geometry	Grid Paper (with Flat Figures)
Calculate the perimeter of a square or rectangle	Geometry	Grid Paper (with Polygon)

4.1 Problem classification

KCs provide a detailed and fine-grained description of individual math skills, as well as in which grade they should be mastered. As such, they offer a convenient summary of both the substantive constraints and the expected difficulty level of individual math skills. Examples are shown in Table 1.

To generate various math problems efficiently, we analyzed the similarities among the KCs and found that the abstract content of different math problems can be clustered in a limited number of patterns. We classified them into five generic templates that represent five abstract problem forms. These templates yield the abstract forms of math problems, using as input the parameters and ranges that encode the KC's requirements and constraints.

4.2 Templates

Table 2 presents the five generic templates for abstract math problem generation, with their definition and the corresponding parameters. These templates generate the abstract math problems using the parameters extracted from the requirements of the corresponding KC. For every template, different settings of the parameters can be used to generate different problem types. The *Equation* template is mainly used for arithmetic problem generation. The equation is encoded as a binary expression tree taking the operands and operators generated from the constraint parameters as the binary nodes, to model the equation and calculate the result. In the *Number sequence* template, the generated number sequence can represent problems such as finding numerical patterns given specific intervals between numbers and using a number line to recognize adjacent numbers; it can generate numeric sorting and comparison problems within any given range. The *Ratio table* template can generate tables modelling various problem types with different attributes and number of rows and columns, such as unit conversion problems, distance-ratio-speed problems, etc. For the *Percentage chart* template, generated abstract percentage charts enable students to practice skills for reading tables and charts, understand part-to-whole relationships, and solving percentage problems. For the *Grid paper* template, the generated figures help students intuitively solve geometric problems, such as measuring the area and perimeter by counting square boxes. When there is more than one geometric shape on the grid paper, it can represent area or perimeter comparison problems.

4.3 Distractor generation

Multiple-choice questions are a traditional method often used to test students' math skills. In them, the correct answer is mixed among a number of incorrect answers, usually called *distractors*. In contrast to other approaches that randomly generate wrong answers, we propose to identify the most common mistakes usually made and encode them in a rule-based distractor generator. This method can be applied in a variety of problem categories, although each automatic distractor generator will typically require category-specific knowledge and rules. For example, common errors in geometric problems may involve confusing angles and measurement directions, while in unit conversion, students might easily get one order of magnitude wrong, or confuse the correct units altogether.

For this project, and as a proof-of-concept of the usefulness of this rule-based approach, we chose to focus only on distractor generation for multiple-choice questions on arithmetic problems, generated using the Equation template. We collected and analyzed student's answers in fill-in-blank arithmetic problems and identified the three most frequent types of wrong answers; see Table 3. Accordingly, we encoded our rule-based method to generate the distractors corresponding to each of those mistake types. In particular, for some multiplication operation, we take advantage of multiplication rules, switching the operands during the

Table 2: *Overview of generic templates*

Template	Definition	Parameters
Equation	Mathematical expression with operands, operators on the left side of the equal sign, and a result number on the right side	1) Operators 2) Range of operands 3) Range of the result
Number Sequence	A sequence of numbers in various data types (including integer, decimal, fraction, time, date, temperature, and money value)	1) Value types 2) Sequence length 3) Range of numbers 4) Specified interval (optional)
Ratio Table	A table of proportional numbers that present the ratio relationship between the specific attributes	1) Ranges of the numbers in the first line 2) Ranges of ratios 3) Attributes types 4) Number of rows and columns
Percentage Chart	A list of totals and percentages, as well as related charts or tables showing them	1) Value type 2) Chart type 3) Number of parts 4) Range of percentages
Grid Paper	Basic geometry including triangle, rectangle, square, and combinatorial polygons, that are presented on a grid paper	1) Geometry type 2) The number of geometric shapes

operation process. For example, for 6×9, the correct result can be calculated by $6 \times 10 - 6$, while the distractor generated performs $6 \times 10 - 9$.

Table 3: *Three common mistakes of arithmetic problems and our designed rules*

Mistake Type	Distractor Generation Rule
Correct operation but wrong calculation	Generate a distractor similar to the correct answer
Correct calculation but using a wrong operation	Replace operator by its inverse, or switch order of operands; in particular, take advantage of operator precedences and disarrange the operation steps
Wrong operation and wrong values	Some other random answer, but within the problem constraints and ranges

5 *Procedural generation of text for math problems*

For text generation, we adapted and integrated the methods of Polozov [9], Koncel [18] and Deane [15]. The text generation process consists of four steps, as depicted in Figure 3: *logic schema generation* from the input abstract math problem, using a database of 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* to detect and fix grammar mistakes. The expression "logic schema generation" refers to constructing a logic representation that encodes the given abstract math

problem. Afterwards, NLG techniques are adopted to realize the structured logic schema into a concrete textual representation.

5.1 Logic schema generation

This step takes an abstract math problem as input, and generates a logic schema of the underlying math problem based on a predetermined ontology. A logic schema is structured by linguistic variables, which include the following elements:

- **Entities**: words annotated with semantic labels, which describe the patterns or categories of vocabulary (e.g. FOOD, VEHICLE and LOCATION, in Table 4).
- **Ontology Relation**: word that defines some relationship between entities (e.g. "HAS" and "SPEND" in Table 4).

Table 4: *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)	Jerry gives Tom 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 at 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 4 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 $(3+2=?, 6\times 2=?)$, 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, various ontology types can be used in the generation of the logic schema. Moreover, the database of ontology types is easily extensible.

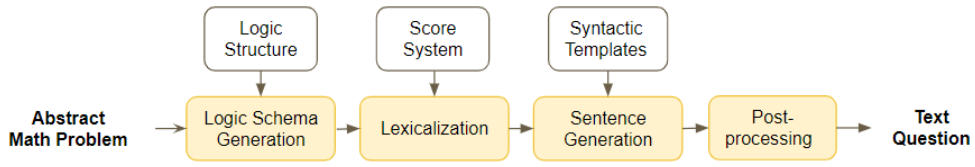

Figure 3: *Main steps of text generation for a math problem*

Therefore, content creators are empowered to design and tweak the ontologies available to steer the logic schema generation.

The ontology constrains the vocabulary used in the sentence, which means there will typically be a limited number of word choices available for the substitution of variables in the logic schema. For example, as shown in Table 4, 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 subsection.

5.2 Lexicalization

The lexicalization step takes the logic schema as input, and assigns actual words to the linguistic variables. The linguistic variables in the logic schema are encoded with specific semantic labels, which should be substituted with suitable words to build a concrete sentence. For each semantic label, there is a word list consisting of the corresponding candidate words. For example, in the fourth logic schema in Table 4, 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 reasonable and coherent text stories, one has to select a plausible and appropriate combination of words. For this, we adopted a scoring method similar to Koncel’s [18], that evaluates the semantic coherence among its candidate words.

This scoring mechanism works as follows: suppose the lexicalization step takes as input the logic schema L, and its linguistic variables $V = v_1, v_2, ..., v_n$ We represent the lexicalization output W in terms of the words that we choose to replace the variables in list V, so $W = w_1, w_2, ..., w_m$; the ontology relation term is also appended to W. We then iteratively choose random words from the word lists to build candidate sets W_k, and we use a score function to determine the best output among them. The score function quantitatively evaluates the semantic coherence among the words based on word embedding similarity provided in WordNet [23]. Formally, we define the scoring function S of a set of words W_k as

$$S(W_k) = \sum_{w_i, w_j \in W_k} cos(w_i, w_j)$$

where $cos(w_i, w_j)$ represents the cosine similarity between the embeddings of two words w_i and w_j. The cosine similarity is usually employed to capture semantic similarity by measuring the angle between two embeddings, and ignoring the vectors’ magnitudes, which may be influenced by extraneous factors, like differences in occurrence frequencies. For the aforementioned example, ("brownie", "cafe") would get a higher score than ("brownie", "playground"). After scoring all candidates, the algorithm selects the word list W with the highest score, i.e. the most semantically coherent.

5.3 Sentence generation

Based on the ontology used in the logic schema generation phase, we designed a variety of syntactic templates, and matched these templates to the logic schema according to their attributes and the structure of ontology relations and entities. This step realizes the lexicalized logic schema into a concrete sentence. The first task in this process is to select a syntactic template that matches the logic schema according to the entities’ constitution and the ontology relation type. Afterwards, the template provides the syntactic structure of the sentence according to a context-free grammar. Syntactic labels, such as VERB and SUBJECT, are replaced with the ontology relations and entities. As a result, a concrete sentence is output.

Table 5 shows an example of a syntactic template that matches a logic schema L. Naturally, these templates are language-specific; in this example we use the English syntactic

Table 5: *A syntatic template matching the example logic schema*

Input Logic schema	BUY(Anna, 6, Brownie, Cafe)
Matched Syntactic Template	SUBJECT VERB NUM OBJ PREP DET NOUN
Output Sentence	Anna buy six brownie at the cafe

template. When a different language version is required, one needs to build and use templates that have the corresponding correct grammar structure (e.g. relative order of subject, predicate, etc.).

5.4 Post-processing

To generate natural language with correct grammar and improve the quality of the text, we designed a post-processing mechanism using the 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 algorithm detects such tags and finds contradictions among them, such as the inconsistent person/tense of the verb and plural mistakes for the nouns, as for example, seen in the output sentence of Table 5; after post-processing, the sentence is turned into "Anna buys six brownies at 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 always 100% accurate due to the nature of NLTK techniques. Therefore, generated text questions should typically be checked, and possibly retouched, to ascertain their full correctness before being offered online.

6 Evaluation

The proposed content generation pipeline aims at supporting, rather than replacing, human content creators, by accelerating their content generation process. Content creators are typically domain specialists (e.g. in math, language or history) who have extensive experience with creating content and teaching it to children. Moreover, they also have abundant experience with young public, so that they can create proper contexts in which to frame abstract (math) problems.

In this section, we describe the two ways in which we assessed the extent to which our approach achieves its goals: (i) assessing how much time our method can spare in the overall content production process, and (ii) measuring the perceived quality of the content generated.

6.1 Time gain in the workflow

Content creators at Squla broadly divide their work in four phases. First, they study the knowledge component for which they create questions, to understand what it does (and does not) cover, as well as its constraints (*KC comprehension*). They then think of an appropriate abstract problem as well as a set of (correct and, possibly, distractor) answers (*Q+A*). Third, they enrich the problem by providing an appropriate context (*contextualizing*). Finally, they look for appropriate imagery (either in the internal database or on iStock) to accompany the question (*image search*). The key objective of this time gain evaluation is to assess if our approach supports content creators through time savings and, possibly, in which phases of this process.

Table 6: *Selected knowledge components*

KC id	KC description	Grade	No. of questions
117	can solve simple addition and subtraction problems in context with total volumes of at least 12	1	3
130	can add and subtract numbers with total volumes of at least 100, in context as well as with formal math language, by using standard procedures like chaining and splitting	2	3
178	can offer a formal multiplication problem when given a contextualized situation, and vice versa	2	2
617	can process information from a description or table into a circle diagram	5	2

Table 7: *Evaluation results regarding the time gain in the content creation workflow*

KC id	Phase	From scratch (min)	From PCG (min)	Time gain (min)	Time gain (%)
117	KC comprehension	5	0	5	100
	Q+A	10	6	4	40
	Contextualizing	10	4.5	5.5	55
	Image search	5	4.5	0.5	10
130	KC comprehension	5	0	5	100
	Q+A	15	6	9	60
	Contextualizing	10	4.5	5.5	55
	Image search	5	4.5	0.5	10
178	KC comprehension	5	0	5	100
	Q+A	5	4	1	20
	Contextualizing	5	3	2	40
	Image search	5	3	2	40
617	KC comprehension	5	0	5	100
	Q+A	10	4	6	60
	Contextualizing	10	3	7	70
	Image search	5	3	2	40

Method To evaluate the potential time gain of our content generation pipeline, we first selected four reasonably different KCs; see Table 6. For each of these KCs, we used our content generation system to generate a number of questions (also indicated in Table 6).

Our plan was to involve a number of Squla's content creators in this evaluation. Unfortunately, due to the company's current internal policy and scarcity of resources, only one of Squla's content creators was made available for this study. She was asked to create two sets of questions for each of these KCs: the first set was created completely from scratch, whereas the second set was created based on the output of our system. In both procedures, the content creator kept track of how much time she spent in each of the four phases indicated above.

Results Table 7 summarizes the results of the time gain evaluation per KC and per phase of the workflow: the total time spent in creating all questions in a KC, both from scratch (third column) and using the procedural content generation approach (fourth column); the absolute time gain (fifth column), and the relative time gain (sixth column).

6.2 Perceived quality of the content

Method To evaluate the quality of the procedurally generated content, a survey was conducted among five Squla's content creators specialized in math content, as well as 12 additional Squla employees who have frequent exposure to existing, human made, content. We asked these 17 participants to evaluate the quality and the usefulness of a number of problems generated by our PCG approach. We classified the problems into two categories: (i) arithmetic problem generated using the Equation template, and (ii) problems in other math topics including ordering, comparison, ratio, percentage, mathematical relationships and geometry.

For the *first category*, we randomly chose *five problems* from a large generated set of arithmetic problems using addition, subtraction, multiplication, division, and combined operations. For these arithmetic problems, in addition to the correct answer, we randomized three generated distractors as alternative answers. The participants were asked to score each arithmetic problem on a 1-5 Likert scale on the following two criteria:

- **Text coherence:** assessing how understandable and sensible is the generated text for primary school students;
- **Quality of distractors:** assessing how appealing and plausible are the generated distractors.

For the *second category*, the survey presented *ten problems* generated by the other templates, i.e. Number Sequence, Ratio Table, Percentage Chart, and Grid Paper. For each template, we sampled two or three typical generated quizzes. Each quiz includes a simple text that intuitively describes the math problem and the related visual content such as a chart, table, or grid paper with figures. Compared to arithmetic problems, the text of these problems is simpler and more straightforward without the need of optimizing phrases and words.

The participants were asked to score all *fifteen problems* (i.e. both the previous five arithmetic problems and these ten other problems), on a 1-5 Likert scale on the criterion:

- **Usefulness:** assessing how well is the generated content aligned with the KCs required by primary school curriculum.

Results In total, from the 17 participants, we received 85 responses (i.e. 17×5 problems) on the first two criteria (text coherence and distractors' quality), and 255 responses (i.e. 17×15 problems) on the third criterion (usefulness). Figure 4 presents the response distribution for the three criteria separately. Moreover, Table 8 summarizes the average scores and the standard deviation for each of the three criteria in the survey.

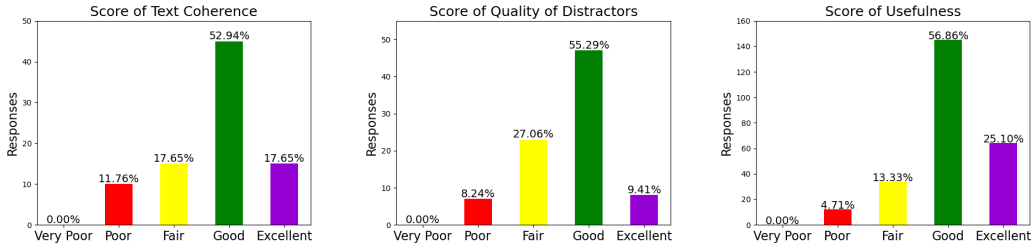

Figure 4: *Score distribution for text coherence, quality of distractors and usefulness*

Table 8: *Overall evaluation results*

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

6.3 Discussion

From the two evaluations performed, the *time gain* analysis is particularly relevant, due to its consequences for the workflow of content creators. As shown in Table 7, the largest workflow time gain arises because the PCG approach renders the *KC comprehension* phase redundant. The reason for this redundancy is not that the content creator had just invested the time to understand the KC, in order to create its questions from scratch. Instead, it is more fundamental: the exact definition and interpretation of each KC has now been recorded once and for all in the KC database of the PCG system (see e.g. Table 1), including all requirements and constraints per KC. As a result, whatever abstract math problem our approach generates for a given KC, fulfilment of all its requirements is automatically guaranteed, and there is no human supervision needed to check that at this stage. Of course, this fact holds regardless of the order of the two sets of questions requested from the content creator. Had we asked a content creator to first create the set of questions using the PCG output of our system, and only afterwards create the second set from scratch, (i) handling the first set would not require nor bring in any comprehension on the details of that specific KC, (ii) that comprehension would still be indispensable for creating the second set and, hence, (iii) the values for KC comprehension time would be similar to those in Table 7. In other words, there is no order bias in this evaluation process. Given the often complex description of the KCs, this time saving alone is quite substantial in the overall content creation process.

Second in line are the time gains created in the *contextualizing* phase, closely followed by the *Q+A* phase, both with a significant time saving as well. Time gains through image search were typically smaller, but still substantial, despite the fact that assisting content creators in the image search phase was not a primary objective of our approach.

All together, the relative time gain of the PCG approach, averaged across all phases of the content creation process, and weighted by the number of questions, amounts to 56%. This is in itself a considerable improvement, which makes the one-time processing of all KCs into the database a very attractive, high-return investment. That is already noteworthy independently of the quality of the content being procedurally generated, which is what our second evaluation investigated.

Regarding the *perceived quality of the content*, the average score for all three criteria is well above 3, as shown in Table 8. This indicates the participants considered that our approach produces content with coherent text and good distractors, useful to help human content editors accelerate their creation process. The standard deviation for the scores of all three criteria is less than 1, indicating that they are mostly centered on scores between 3 and 5, which means most participants are satisfied with the generated content.

Regarding *text coherence*, the survey reveals that the output of our text generation pipeline was considered as coherent text questions with sensible context. As shown in Figure 4, the majority of responses (≈53%) score the text coherence as Good, and ≈18% of the responses even flagged it as Excellent. In order to better understand these scores, we have included an open question asking for a short explanation in case of a lower score. The most valuable

answers mentioned that (i) the usage of some words and ontology was not as sensible and fluent as in human created items, and (ii) the narrative sometimes lacked a more detailed explanation of the context. Both issues can be tackled by exploring more careful and elaborate vocabulary and ontology structures; this was beyond the context of this project, but it should definitely be done when rolling out this approach in the content production workflow. Finally, it is also possible that there is some slight bias in the survey results, as all participants knew in advance that the textual math problems they were evaluating had been procedurally generated, rather than manually created by human experts.

The survey results regarding the *quality of distractors*, with 55% responses flagged as Good and 9% as Excellent, indicate that participants found our rule-based method able to create distractive and plausible wrong answers. As our designed rules were based on frequent mistakes that children typically make, the generated distractors are not as tailored as human-authored distractors created for each specific problem. Fortunately, our rule-based distractor generator is very modular and easily expansible.

Together with the arithmetic problems, the problems generated for other knowledge topics were also found to be useful for our target users. Globally, the majority of responses (≈57%) score the *usefulness* of the generated problems as Good, with an additional ≈25% that found it Excellent. There is no clear difference between the usefulness score for arithmetic problems and for non-arithmetic ones, as most of these problems proved to be quite well-aligned with the curriculum and satisfy the primary school requirements.

All in all, the various text questions generated by our logic schema generation and optimized lexicalization methods were quite positively received by the target audience. The concept of knowledge component, with its constraints, is instrumental for the selection and steering of abstract math problem templates in our approach. Compared to neural network-based methods [22], it effectively provides content creators much better control over the difficulty level of the output problems. In addition, because our text generation approach directly seeks semantic coherence, content creators can always input new logic schemas and enrich the word vocabulary to generate novel text questions. This significantly improves the quality and variety of the generated text content, compared to that of standard template-based text generation methods [15].

Summarizing, the survey results seem to confirm that our generic template-based abstract problem generation method succeeds in generating a variety of math problems about the desired KCs and with constrained difficulty levels, and that this content is found understandable, sensible and useful for human content creators.

7 Conclusion

Creation of quality content for online educational services is increasingly important, but conventional manual processes are extremely labor-intensive and expensive. In this paper we investigated how the application of PCG techniques for math problem generation can alleviate that, and identified the strengths and weaknesses of the most promising methods developed so far.

We proposed a novel approach to procedurally generate math problems, integrating a template-based abstract problem generator that allows human content creators to configure their generation, based on the curriculum requirements expressed by so-called knowledge components, which include the desired constraints and difficulty level. Moreover, our approach features a text generation pipeline that produces coherent text questions by integrating natural language generation with an optimized word selection method.

Evaluation by human experts concluded that our approach introduces substantial time

savings in all phases of the content creation process, with an average workflow time gain of 56%. In addition, our method produces solvable math problems that are considered sensible for primary school students and very useful for content creators. A major strength of this approach is the controllability and flexibility of the templates used: human content editors, even with no programming skills, can easily control and modify the parameters and variables in the abstract templates, as well as configure and extend the vocabulary and ontology types for the text generation step, based e.g. on specific curriculum requirements.

This approach can be extended to generate math problems in more knowledge topics by adding new templates. Furthermore, it would be interesting to explore and introduce AI-based methods to automatically generate the ontology used in the text generation process [24]. Last but not least, the integration of this approach with a convenient (student) player model has the potential to significantly improve the overall personal learning experience [3]: not only would it guarantee that the adaptive content being generated fits the skills and needs of each individual student, but it would also allow for a fine customization of this content to their personal preferences.

Acknowledgements

We thank Peter Hofstede, Linda Rijpert, Nienke Post and Pauline Rambonnet for their support and feedback at various stages of this project. We also thank all Squla colleagues who participated in our evaluation, for their valuable feedback. Finally, we thank Martin Skrodzki for his sharp comments on a previous version of the manuscript.

References

[1] R. Lopes, K. Hilf, L. Jayapalan, and R. Bidarra, "Gameplay semantics for authoring adaptivity in mobile games," in *Proceedings of FDG 2013 - Eighth International Conference on the Foundations of Digital Games*, Chania, Crete, Greece, 2013.

[2] J. Zhu and S. Ontañón, "Player-centered AI for automatic game personalization: Open problems," in *International Conference on the Foundations of Digital Games*. Association for Computing Machinery, 2020. doi: 10.1145/3402942.3402951

[3] R. Lopes and R. Bidarra, "Adaptivity challenges in games and simulations: a survey," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 2, pp. 85–99, 2011, doi: 10.1109/TCIAIG.2011.2152841.

[4] N. Shaker, J. Togelius, and M. J. Nelson, *Procedural content generation in games*. Springer, 2016, doi: 10.1007/978-3-319-42716-4.

[5] E. Rijnboutt, O. Hokke, R. Kooij, and R. Bidarra, "A robust throw detection library for mobile games," in *Proceedings of FDG 2013 - Eighth International Conference on the Foundations of Digital Games*, Chania, Crete, Greece, 2013.

[6] R. M. Smelik, T. Tutenel, R. Bidarra, and B. Benes, "A survey on procedural modeling for virtual worlds," *Computer Graphics Forum*, vol. 33, no. 6, pp. 31–50, 2014, doi: 10.1111/cgf.12276.

[7] A. M. Santos, P. A. Santos, F. M. Dionisio, and P. Duarte, "On-line assessment in undergraduate mathematics," in *Proceedings of 2nd International Conference on the Teaching of Mathematics*, Crete, Greece, 2002.

[8] G. Smith and C. Harteveld, "Procedural content generation as an opportunity to foster collaborative mindful learning," in *Workshop on Games and Learning, co-located with International Conference on the Foundations of Digital Games*, 01 2014.

[9] O. Polozov, E. O'Rourke, A. M. Smith, L. Zettlemoyer, S. Gulwani, and Z. Popović, "Personalized mathematical word problem generation," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015. [Online]. Available: https://www.ijcai.org/Proceedings/15/Papers/060.pdf

[10] A. M. Smith, E. Andersen, M. Mateas, and Z. Popović, "A case study of expressively constrainable level design automation tools for a puzzle game," in *Proceedings of the Seventh International Conference on the Foundations of Digital Games*. ACM, 2012, pp. 156–163, doi: 10.1145/2282338.2282370.

[11] D. Hooshyar, M. Yousefi, M. Wang, and H. Lim, "A data-driven procedural-content-generation approach for educational games," *Journal of Computer Assisted Learning*, vol. 34, no. 6, pp. 731–739, 2018. doi: 10.1111/jcal.12280. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/jcal.12280

[12] R. Singhal, M. Henz, and K. McGee, "Automated generation of geometry questions for high school mathematics," *CSEDU 2014 - Proceedings of the 6th International Conference on Computer Supported Education*, vol. 2, pp. 14–25, 01 2014.

[13] L. Rodrigues, R. P. Bonidia, and J. D. Brancher, "A math educacional computer game using procedural content generation," in *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, vol. 28, no. 1, 2017, p. 756, doi: 10.5753/cbie.sbie.2017.756.

[14] N. A. Khodeir, H. Elazhary, and N. Wanas, "Generating story problems via controlled parameters in a web-based intelligent tutoring system," *The International Journal of Information and Learning Technology*, vol. 35, no. 3, pp. 199–216, 2018, doi: 10.1108/IJILT-09-2017-0085.

[15] P. Deane and K. Sheehan, "Automatic item generation via frame semantics: Natural language generation of math word problems." 2003. [Online]. Available: https://eric.ed.gov/?id=ED480135

[16] B. Wyse and P. Piwek, "Generating questions from openlearn study units," in *AIED 2009 Workshop Proceedings Volume 1: The 2nd Workshop on Question Generation*, 2009. [Online]. Available: http://oro.open.ac.uk/32435/

[17] M. Gupta, N. Gantayat, and R. Sindhgatta, "Intelligent math tutor: Problem-based approach to create cognizance," in *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale*. Association for Computing Machinery, 2017, pp. 241—-244, doi: 10.1145/3051457.3053995.

[18] R. Koncel-Kedziorski, I. Konstas, L. Zettlemoyer, and H. Hajishirzi, "A theme-rewriting approach for generating algebra word problems," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2016, pp. 1617—1628, doi: 10.18653/v1/D16-1168.

[19] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 3111–3119.

[20] K. Orkphol and W. Yang, "Word sense disambiguation using cosine similarity collaborates with word2vec and WordNet," *Future Internet*, vol. 11, no. 5, p. 114, 2019, doi: 10.3390/fi11050114.

[21] K. Nandhini and S. R. Balasundaram, "Techniques for generating math word questions for learning disabilities," in *IJCA Proceedings on International Conference and Workshop on Emerging Trends in Technology (ICWET)*, 2011, pp. 14–25.

[22] V. Liyanage and S. Ranathunga, "A multi-language platform for generating algebraic mathematical word problems," in *2019 14th Conference on Industrial and Information Systems (ICIIS)*. IEEE, 2019, pp. 332–337, doi: 10.1109/ICIIS47346.2019.9063354.

[23] G. A. Miller, “WordNet: a lexical database for english,” *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995, doi: 10.1145/219717.219748.

[24] J. Zhu, A. Liapis, S. Risi, R. Bidarra, and G. M. Youngblood, “Explainable AI for designers: a human-centered perspective on mixed-initiative co-creation,” in *Proceedings of CIG 2018 - IEEE Conference on Computational Intelligence and Games*, 2018, doi: 10.1109/CIG.2018.8490433.