# Procedural generation of problems for elementary math education

**Hoe zou je het aanbod van wiskundige content in de game (denk aan oefeningen, tips en uitleg) beter kunnen afstemmen op de actuele kennis van een leerling?**

Wat voor soort AI componenten, tools en technieken (zoals genoemd in sectie 4 van Westera et al. (2020), de andere drie artikelen, en evt. uit je eigen ervaringen met AI-tools) zouden ingezet kunnen worden om de game interessanter, adaptiever, en/of effectiever te maken, en waarom zou je daarvoor kiezen?

Welk probleem wordt hiermee aangepakt? Wie wordt daardoor waarin ondersteund?

Hoe zou je oplossing eruit kunnen zien? Maak een paper prototype van tenminste een scherm dat een indruk geeft van de nieuwe functionaliteit en interactie.

Wat is daarvoor nodig, qua data? Geef een voorbeeld van hoe die data eruit zou kunnen/moeten zien.

## Introduction

Onlineonderwijs is steeds gebruikelijker helemaal sinds corona. Het creëren van wiskundige vragen voor deze games is arbeidsintensief, tijdrovend en duur. Het maken van één nieuwe vraag kan meer dan 100 euro kosten. Dit wordt helemaal een probleem bij adaptieve gameplay, waarbij de moeilijkheidsgraad automatisch aanpast aan de speler. Hier zijn namelijk heel veel vragen voor nodig. In plaats van handmatige creatie kan er gebruik gemaakt worden van **Procedural Content Generation (PCG).** Dit richt zich op het automatisch genereren van content via algoritmische methoden, met beperkte of indirecte menselijke tussenkomst. Onderzoek toont aan dat PCG het proces van contentcreatie aanzienlijk kan versnellen en zelfs de variëteit en originaliteit ervan kan vergroten.

Er zijn een aantal uitdagingen binnen dit onderwerp die moeten worden aangepast. Allereerst moeten menselijke contentmakers voldoende controle hebben over de contentgenerator (bijvoorbeeld wat betreft thema’s, objecten en gebruikte afbeeldingen). Ten tweede moet het **PCG-proces** tekstuele content genereren zonder concessies te doen aan kwaliteit en variatie. Ten derde moet het PCG-proces aansluiten bij de dynamische vereisten van contentonderwerpen en moeilijkheidsgraden, die mogelijk ook afhangen van het profiel van de leerling (bijvoorbeeld leeftijd, vaardigheden, etc.).

Er is een procedurele generatiepijplijn ontworpen voor wiskundecontent die wiskundeopgaven kan genereren met zowel een tekstuele als visuele representatie, terwijl slechts één invoer nodig is: een zogenaamde **"knowledge component"**, die de vereisten vastlegt op basis van zowel het onderwerp als het moeilijkheidsniveau.

## Related work

### PCG in math education

PCG zijn algoritme die digitale content creëren met gelimiteerd of indirect gebruikersinput en wordt gebruikt in het ontwerpen van games, puzzels en virtual environments. In de afgelopen jaren worden PCG ook gebruikt om content voor wiskundeonderwijs te creëren. Het is niet allen nuttig voor het creëren voor betrouwbare en aanpasbare game content maar zorgt ook voor discussie over de rede van het antwoord in plaats van het antwoord zelf. Zie paragraaf voor voorbeelden van PCG in wiskunde.

### Natural Language Generation for Math Problems

NLG is het proces van het omzetten van gestructureerde gegevens in natuurlijke, verhalende taal. Er is verschillend onderzoek gedaan om dit in te zetten voor het genereren van verhalen voor wiskunde problemen. Zie paragraaf voor voorbeelden. Echter hebben deze oplossingen nog beperkingen.

## General approach

Het SLO heeft een framework voor het basisonderwijs in wiskunde, inclusief de beschrijving van honderden leerdoelen, aangeduid als kenniscomponenten. In het paper ontwikkelen zij een pipeline die als input de gewenste kenniscomponent heeft. De output is een quiz bestaande uit:  
(i) een abstract wiskundeprobleem dat overeenkomt met die KC,  
(ii) de tekstuele en visuele uitwerking van het probleem,  
(iii) het juiste antwoord en, indien van toepassing,  
(iv) zogenaamde ‘afleiders’ (indien het een meerkeuzequiz betreft).

Als eerste wordt het abstracte wiskundeprobleem gecreëerd vervolgens de overige onderdelen.

Wiskunde vragen worden afzonderlijk gecreëerd van tekst en visuele inhoud. Hierdoor kan de generator efficiënt een groot aantal problemen genereren met controleerbare moeilijkheidsniveaus en gewenste probleemtypen.

Voor het genereren van abstracte wiskundeproblemen stellen we een sjabloongebaseerde methode voor, waarbij vijf generieke, geparametriseerde abstracte templates worden gebruikt. De parameters en variabelen van deze abstracte templates stellen ons in staat om de moeilijkheidsgraad en eigenschappen van de wiskundeproblemen te sturen. Hebben nog geen relatie met tekst.

In de fase van tekstopwekking wordt het abstracte probleem als input gebruikt en omgezet in een tekstuele vraag in natuurlijke taal. Worden er een combinatie van NLG-technieken gebruikt.

Voor de generatie van antwoorden in meerkeuzevragen stellen we een regelgebaseerde methode voor om misleidende maar plausibele foute antwoorden (‘afleiders’) te genereren op basis van het gegeven abstracte probleem. Daarnaast hebben we een eenvoudig visueel inhoudsmodel ontwikkeld dat automatisch relevante afbeeldingen uit een grote database haalt, thematisch passend bij de tekstuele representatie van het gegenereerde abstracte probleem.

## Procedural generation of abstract math problems

De input bestaat uit **kenniscomponenten**, die elk een onderwerp (bijv. **rekenkunde**), vereisten (bijv. **optellen**) en beperkingen (bijv. **binnen 100**) hebben. Een ingevoerde kenniscomponent (KC) wordt eerst geclassificeerd om een geschikt sjabloon te selecteren voor de generatie van het abstracte wiskundeprobleem. Vervolgens genereert de pijplijn automatisch het juiste antwoord en, indien van toepassing, de **afleiders** (foutieve, maar plausibele antwoorden).

### Problem classification

Ze hebben deze kc geclassificeerd in **vijf generieke** templates, die vijf abstracte probleemvormen vertegenwoordigen. Deze templates leveren de abstracte vormen van wiskundeproblemen op, waarbij als input de parameters en bereiken worden gebruikt die de vereisten en beperkingen van de KC coderen.

### Templates

Deze templates genereren abstracte wiskundeproblemen op basis van de parameters die zijn afgeleid uit de vereisten van de bijbehorende kenniscomponent (KC). Voor elk sjabloon kunnen verschillende parameterinstellingen worden gebruikt om verschillende soorten problemen te genereren.

### Distractor generation

Bij meerkeuze fouten antwoorden genereren door de meest voorkomende fouten die leerlingen maken te identificeren en deze te coderen in een **regel gebaseerde afleidergenerator**. De antwoorden van leerlingen op invulvragen met rekenkundige problemen zijn verzameld en geanalyseerd en de drie meest voorkomende fouttypen zijn geïdentificeerd. Op basis hiervan hebben zijn er regel gebaseerde methode gecodeerd om afleiders te genereren die overeenkomen met elk van deze fouttypen.

## Procedural generation of text for math problems

Het proces van tekst generatie bestaat uit **vier stappen:**

1. **logic schema generation** op basis van het ingevoerde abstracte wiskundeprobleem, met behulp van een database met ontologische relaties en entiteiten.

2. **Lexicalisatie**, waarbij de semantische configuraties worden omgezet in samenhangende woorden met behulp van een beoordelingssysteem.

3. **Zinsgeneratie** op basis van generieke, taalafhankelijke syntactische sjablonen.

4. **Nabewerking** om grammaticale fouten op te sporen en te corrigeren.

### Logic schema generation

De input is een abstract wiskundeprobleem en de output is een l**ogisch schema** van het onderliggende wiskundeprobleem op basis van een vooraf bepaalde **ontologie**.

Een logisch schema is gestructureerd met **linguïstische variabelen**, die de volgende elementen bevatten:

* **Entiteiten**: woorden die zijn geannoteerd met **semantische labels**, die de patronen of categorieën van de woordenschat beschrijven (bijv. **FOOD, VEHICLE** en **LOCATION** in **Tabel 4**).
* **Ontologische relaties**: woorden die een **relatie** tussen entiteiten definiëren (bijv. **"HAS"** en **"SPEND"** in **Tabel 4**).

### Lexicalization

De lexicalization neemt als input het logic schema en wijst werkelijke worden toe aan linguïstische variabelen. De linguïstische variabelen in het logisch schema zijn voorzien van **specifieke semantische labels**, die moeten worden vervangen door geschikte woorden om een concrete zin te vormen. Voorbeeld “FOOD” door *"brownie"*, *"pizza"*, *"cookie"*, *"wafel"*, enz.

Het scoremechanisme werkt als volgt:

1. **Input**: Het lexicalisatieproces ontvangt het logisch schema **L**, en de lijst met linguïstische variabelen **V = {v1, v2, ..., vn}**.
2. **Kandidaatselectie**: De output van het lexicalisatieproces **W** bestaat uit de woorden die de variabelen in **V** vervangen, dus **W = {w1, w2, ..., wm}**. De ontologische relatie wordt hieraan toegevoegd.
3. **Scoring**: Willekeurige combinaties van woorden worden gekozen uit de woordenlijsten om **kandidaatsets Wk** te vormen. Vervolgens wordt een **scorefunctie** gebruikt om de beste output te bepalen.
4. **Cosinusovereenkomst**: De scorefunctie **S(Wk)** evalueert de **semantische samenhang** tussen de woorden op basis van de **woordembedding-overeenkomst** in **WordNet [23]**.

### Sentence generation

Op basis van de **ontologie** die is gebruikt in de **logisch-schema-generatiefase**, hebben we een reeks **syntactische sjablonen** ontworpen. Deze sjablonen worden gekoppeld aan het logisch schema op basis van hun kenmerken en de **structuur van ontologische relaties en entiteiten**.

Deze stap zet het **gelexicaliseerde logisch schema** om in een **concrete zin**.

**Proces van syntactische generatie**

1. **Selectie van een syntactisch sjabloon**
   * Het sjabloon wordt gekozen op basis van de **opbouw van de entiteiten** en het **type ontologische relatie** in het logisch schema.
2. **Structurering van de zin**
   * Het sjabloon bepaalt de **syntactische structuur** van de zin volgens een **contextvrije grammatica**.
   * Syntactische labels zoals **VERB** en **SUBJECT** worden vervangen door de relevante **ontologische relaties en entiteiten**.
3. **Generatie van de uiteindelijke zin**
   * Dit resulteert in een **concrete, grammaticaal correcte zin**.

### Post-processing

Om geen grammatica fouten te genereren is er een post-processing mechanisme ontwikkeld met de **Natural Language Toolkit (NLTK)** en de **Part-of-Speech (POS) tagging-techniek**.

De **POS-tagger** kan in een zin de **onderwerpen, werkwoorden, getallen, eenheden, objecten en bijvoeglijke naamwoorden** identificeren. Ons algoritme detecteert deze **tags** en spoort **tegenstrijdigheden** op, zoals:

* **Inconsistente persoons- of tijdsvormen** van het werkwoord
* **Meervoudsfouten** bij zelfstandige naamwoorden

Echter, de **output is niet altijd 100% accuraat** vanwege de **beperkingen van de NLTK-technieken**. Daarom is het aanbevolen dat de **gegenereerde tekstvragen** worden **nagekeken en eventueel aangepast** voordat ze **online worden gepubliceerd**.

## Evaluation

De pipeline is bedoeld om vragen makers te ondersteunen en niet te vervangen. Het is bedoeld om tijd te besparen.

### Time gain in the workflow

Vragen creëren zijn onderverdeeld in 4 categorieën in elke categorie tijdswinst.

### Perceived quality of the content