UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS/IT Honours
Final Paper 2020

Generating Natural Language IsiZulu Text From Mathematical Expressions

Author: Shannon Smith

Project Abbreviation: MATHVERB

Supervisor(s): Zola Mahlaza

Category	*Min*	*Max*	Chosen
Requirement Analysis and Design	*0*	*20*	0
Theoretical Analysis	*0*	*25*	0
Experiment Design and Execution	*0*	*20*	20
System Development and Implementation	*0*	*20*	5
Results, Findings and Conclusions	*10*	*20*	20
Aim Formulation and Background Work	*10*	*15*	15
Quality of Paper Writing and Presentation	*10*		10
Quality of Deliverables	*10*		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	*0*	*10*	0
Total marks		**80**	

Generating Natural Language IsiZulu Text From Mathematical Expressions

Shannon Smith
Department of Computer Science
University of Cape Town
smtsha028@myuct.ac.za

ABSTRACT

A screen reader is an assistive technology that helps visually impaired people to read electronic text. Screen readers audibly render mathematical expressions in their markup language format, which is difficult for listeners to understand. We want to join the worldwide effort in improving math accessibility for people with visual impairments, by focusing on isiZulu speakers. To improve access to mathematics, we investigate how to build a natural language generation system, that uses template-based realisation combined with a few word-level grammar rules, to generate textual descriptions of math expressions. We then conduct a questionnaire with isiZulu speakers to evaluate the understandability of this generated text. Based on the results of the questionnaire, simple and complex descriptions are perceived as understandable on average for an isiZulu speaker; but they can only correctly understand the formula 50% of the time. This gap in perceived understanding and actual understanding was found to be caused by errors in the square root, integral and minus templates. In conclusion, we demonstrated that a template-based NLG system can translate mathematical expressions into natural language isiZulu text, that is perceived to be understandable on average.

1 INTRODUCTION

There is a global effort to make textbooks available online for visually impaired people, where the digital text can be processed by screen readers. Screen readers, a form of assistive technology, use a text-to-speech (TTS) system to convert electronic text into synthesised speech. There are efforts to make this possible for SA as the South African Centre for Digital Language Resources (SADiLaR) supports research on all aspects of natural language processing – including TTS systems, automatic speech recognition, and much more. The TTS systems in their repository can convert text into synthesised speech for most of South Africa's 11 official languages.

Despite this, accessing mathematics through digital text still presents a challenge for people with visual impairments because these TTS systems cannot produce easily understood descriptions of mathematical formulae, leaving users unable to efficiently listen to mathematics. While mathematical expressions formatted in LaTeX, MathML, or other markup languages can be processed by these TTS systems, the TTS system renders the expression in its markup language format. Listening to mathematical expressions in a markup language is inefficient for several reasons; It requires some knowledge of that language, it is slow to listen to, and it sounds unnatural.

A possible solution to this problem could be natural language generation (NLG), as demonstrated by (Ferres and Sepúlveda, 2011; Mazzei et al., 2019; Mondal and Jawahar, 2019). NLG is a subfield of computational linguistics and artificial intelligence that maps some input data to readable, natural language text (Reiter and Dale, 1997). The input data is usually in a non-linguistic format, like numerical data, graphs, images, or in the context of mathematics, MathML and LaTeX; and the output is always human readable text. The process of translating mathematics into natural language text is called math verbalisation, a process which could make expressions more understandable when read aloud by a TTS system than its markup language equivalent. For this reason, an NLG system was implemented.

When deploying an NLG system in such a scenario, the math expressions can be captured using MathML, the World Wide Web Consortium's (W3C) standard for encoding the presentation of formulae and embedding mathematics in webpages. This format has played a key role in math accessibility; specifically, speech synthesis and producing natural language text (MathPlayer (Soiffer, 2005), VoiceXML (Reddy et al., 2005) , and AudioMath (Ferreira and Freitas, 2004). We investigate a MathML-to-Text system that specifically captures mathematical expressions in a Content MathML format.

In this study, we aim to improve access to mathematics for isiZulu speakers with visual impairments, through one research question: How do we build a MathML-to-Text NLG system whose output is perceived as understandable by isiZulu speakers? This project cannot tackle all 11 official languages, so it focuses only on isiZulu. As far as we know, there are no systems in place to verbalise mathematical expressions in isiZulu, in South Africa. Although isiZulu is spoken by 23% of the population, it is still under-resourced in software applications (Grover et al., 2011). The biggest barrier for developing Language Technology for isiZulu is the lack of resources and the complexity of its morphology. Therefore, we hope this project will contribute to improving resources for otherwise low-resourced languages in South Africa.

To evaluate the perceived understandability of this MathML-to-Text system's output, we invite isiZulu speakers to participate in a questionnaire that collects data on their understandability of several generated descriptions.

The paper is structured as follows. In Section 2, we give a brief survey of the current state of NLG and math accessibility in the world. In Section 3, we describe the methods used to answer the research question. In Section 4, we evaluate the understandability of our verbaliser's output by performing a human-based evaluation with isiZulu speakers, and present the results in Section 5. Section 6 presents the discussion of results and finally Section 7 concludes this study.

2 BACKGROUND AND RELATED WORK

We introduce natural language generation (NLG) with a particular emphasis on the concept of linguistic realisation. We discuss the structure of the isiZulu grammar and what method of realisation suits its morphology. We then review a set of works that are related to our problem domain, the access of mathematics for visually impaired people, and finish off with the NLG techniques that have been used in math verbalisers for other languages.

2. 1 Linguistic Realisation

A typical NLG system includes the same six tasks proposed by (Reiter and Dale, 1997); **Content determination**: Deciding what data to include in the final text. **Document structuring**: the overall organisation of the information included in the text. **Aggregation:** combining similar sentences to improve human readability. **Lexical choice:** selecting appropriate words to describe the data. **Referring expression generation:** Selecting words or phrases to identify domain entities. **Linguistic Realisation:** also called Realisation, is the final step of generating a complete sentence after choosing the appropriate words and phrases. This entails filling in any gaps that were not present in the input, such as adding articles, prepositions, auxiliary verbs, and punctuation. The realiser ensures agreement of nouns, verbs, and tenses, by following some form of morphological rules. There are several methods of realisation that we will discuss: templates, grammar-based systems, data-driven approaches and grammar-infused templates.

Templates are an early realisation technique that involves filling in the slots of a predefined sentence (Gatt and Krahmer, 2018). An example of a template for a basic addition operator is: <$OP1 plus $OP2>, where OP1 and OP2 are substituted for any operand, and can generate a sentence such as, "X plus Y".

Grammar-based systems construct sentences based on the grammar rules of the language being used (Gatt and Krahmer, 2018). These systems require a great deal of detail specific to the language, therefore, they do not work well as "plug-and-play" modules. This issue has prompted the use of realisation engines that give users morphology APIs, like SimpleNLG and Grammatical Framework (GF). Although grammar-based systems may be more challenging to develop, they have a few advantages over templates; they are domain independent and the quality of the output is generally higher than templates.

Alternatively, realisation can be accomplished using data-driven approaches, such as training a model using 'end-to-end' machine learning (Gatt and Krahmer, 2018).

Lastly, there is another type of template-based system that uses traditional templates with some grammar rules on top of them (introduced as grammar-infused templates by (Mahlaza and Keet, 2019)). These rules can be word-level functions on a template, that ensure grammatically correct text without having to implement a full grammar engine. This combines the benefits of both approaches – templates are simple to construct but lack the ability to handle complex grammars; full grammar systems are expensive to develop but can accommodate grammar complexity.

However, realisation techniques are language dependent, and are not suitable for every language. Templates and grammar engines are suited to Indo-European languages but fail with isiZulu's agglutinating grammar. Unlike Indo-European languages, which express the tense, negation, etc. in separate components, agglutinative languages use prefixes and suffixes.

In isiZulu, there are a total of 17 noun classes, with a single and plural form in each class. The complex verb is extended with prefixes that reflect subject and object concords, tense/aspect, mood, and negation. On top of the prefixes, there are suffixes that represent stative, applicative, reciprocal, and passive verbs. Further, the verb ends with a final vowel that is dependent on negation. The complex verb structure with all its affixes is shown in (1).

(1) <NEG> <SC> <T/A> <MD> <OC> <VR> <Extension><FV>

Where, NEG=negative, SC=subject concord, T/A=tense/aspect, MD=mood, OC=object concord, VR=verb root, FV=final vowel.

To illustrate why traditional templates are not suited for isiZulu; a common approach to NLG is the use of templates with a few simple rules for template selection. For an English system, it may require a rule to select a template with the article 'an' instead of 'a' if the object noun starts with a vowel. For an isiZulu system, there would have to be a rule for how each of the 17 noun classes affects the verb. This complexity makes the use of templates on their own infeasible (Keet and Khumalo, 2017). Similarly, realisation engines, like SimpleNLG and Grammatical Framework, are encoded for grammatical structures close to English. Hence, there has not been any implementations of isiZulu applications using grammar engines.

A data-driven approach to NLG requires a comprehensive data resource (such as a large, detailed corpus), which is not applicable for under resourced computational languages like isiZulu.

Instead, grammar-infused templates can be used, as shown by (Keet and Khumalo, 2017) for a different domain. This is the hybrid realisation method that our system will use to handle prefixes and suffixes.

2.2 Math Verbalisers

There has been significant research done on improving math accessibility for visually impaired people. We note that there are two strains of technologies that provide access to mathematics: those that translate MathML or LaTeX expressions into Braille,

such as the LAMBDA project (Bernareggi, et al, 2006); and those that render speech or natural language text. The following papers address our problem domain using the latter approach.

AsTeR (Raman, 1994), is an early project from the 90s that helps produce rendered audio of digital documents. This system takes a LaTeX document and parses it into a system that generates natural language text based on rules, called Audio Formatting Language (AFL). Another paper that assists with accessing LaTeX formatted mathematics in PDF documents is Axessibility (Ahmetovic et al., 2018). Axessibility, is a LaTeX package that automatically generates alternative text descriptions of expressions and stores them in a hidden comment attached to the expression. MathPlayer (Soiffer, 2005) is a plug-in for Microsoft Internet Explorer that renders a visualisation of MathML, which allows people with visual impairments (blindness, low-vision and learning disabilities) to dynamically interact with mathematical expressions. MathPlayer integrates with screen-readers and supports synthesised speech generation.

There are math verbalisers that explicitly use the NLG realisation methods discussed in Section 2.1 to generate textual descriptions. The relevant details of these systems are shown in Table 1.

Table 1: Math Verbalisers that Implement NLG Realisation Methods

Paper	Output Language	Realisation Method
Ferres and Sepúlveda, 2011	Spanish	Template-based
Mazzei et al, 2019	Italian	Grammar Engine, SimpleNLG
Mondal and Jawahar, 2019	English	Data-driven methods

(Ferres and Sepúlveda, 2011) approached the problem of math verbalisation in Spanish with a template-based NLG system called MathAcc. Mathematical expressions found on Wikipedia are rasterised images of LaTeX expressions, which screen readers cannot process. To tackle this problem, (Ferres and Sepúlveda, 2011) collected LaTeX expressions from Wikipedia repositories, translated them into a more explicit format, Content MathML (CMML), and generated textual descriptions from the CMML expressions. The descriptions were then embedded into the Wikipedia pages.
In comparison, (Mazzei et al, 2019) achieved similar goals using the grammar engine, SimpleNLG, to generate Italian descriptions of CMML expressions, which were then processed by a TTS system.
(Mondal and Jawahar, 2019) proposed a data-driven approach to handle the issue of TTS systems' inability to read document images of math expressions. They developed a trainable deep neutral network, based off image captioning techniques. The model reads mathematical formulae as document images and then learns to generate appropriate textual descriptions.

The works discussed in math verbalisation have used realisation techniques that are not applicable to isiZulu's complex grammar. To our best knowledge there are currently no MathML-to-Text NLG systems for isiZulu, thus, this paper investigates how we can accomplish building an understandable MathML-to-Text translator, using an appropriate realisation method.

3 BUILDING THE MATH VERBALISER

We follow Dale and Reiter's typical NLG tasks by building an NLG pipeline. This pipeline processes CMML expressions and uses templates with a few embedded functions to generate text. To create these templates, we require building a small corpus of example sentences that the templates should be able to generate.
To ensure project feasibility, we selected eight mathematical operations to verbalise. This selection was based on how frequently the operator is used. (Ferres and Sepúlveda, 2011) found that the most popular operators found in Wikipedia repositories are divide, power (superscript), equations, integrals, addition, subtraction, square root, multiplication. Hence, these are the eight operators chosen.

The following subsections go into detail about how the corpus was created and analysed to form templates for each operator. Then finally, how the NLG system was implemented.

3.1 Corpus Creation

Before the verbaliser could be implemented, we needed to establish what the system was expected to produce. We took a corpus-based approach (Reiter and Dale, 1997) and asked a domain expert to write out examples of appropriate output texts, given some input expressions. In this case, the domain expert was a mathematician who speaks isiZulu.

The aim of gathering these example outputs was to establish: what is the correct wording; how affixes change for each operator/operand; how punctuation is used to break up operators in a sentence. There is an issue of ambiguity in the order of operators when reading the expression out loud. $\frac{(x+y)}{z}$ is read aloud as "x plus y over z". To a listener, this could be misinterpreted as $(x+\frac{y}{z})$. This was an important factor when consulting with the expert – we needed to find a natural way to enforce precedence between operators.

Seventy (70) example mathematical formulae were written out in a PDF file, with roughly 8 examples per operator, and the expert was asked to describe each formula in isiZulu natural language. Most examples were kept simple, containing a single operator and the two operands it worked on. There were also 'combination' expressions, these examples were more complex (containing 2 or more operators), to see how grammar and punctuation plays a part in operator precedence.
The corpus was iteratively created – with multiple collaborations before the final version was agreed upon. Together with the expert, we built a collection of 65 input-output pairs. Five

expressions could not be completed by the expert and those were not included in the collection.

3.2 Template Design

After analysing this corpus data, we could manually design templates for the verbaliser. We discovered that when numeric operands were kept as Arabic numerals, there was a clear pattern that every description followed. We converted this pattern into a general template, called the 'primary template' (1):

(1) U-$operand1 <verb> <object prefix>-$operand2

The template's text includes a single operator (the verb), the object prefix, and two slots for the operands (except in the case of square root and integration, which operate on a single operand). Mathematical operators are inherently nested, operators act upon other operators, and the isiZulu verb prefix changes depending on where it is placed in the sentence. We handled this with a 'secondary template', for situations where a template had to be substituted into a 'primary template'. These secondary templates are created with an embedded function that changes the primary verb into a secondary verb. The primary and secondary templates are compared for three operators in Table 2.

Table 2: The Primary and Secondary Templates for Power, Times and Plus Operators.

Operator	The (1) Primary and (2) Secondary Templates
Power	(1) U-$OP1 ukhuliswe ngenkombananikuphindwa u-$OP2
	(2) $OP1 okhuliswe ngenkombananikuphindwa u-$OP2
Times	(1) U-$OP1 simphindaphinda ngo-$OP2
	(2) OP1 esimphindaphinda ngo-$OP2
Plus	(1) U-$OP1 simhlanganisa no-$OP2
	(2) $OP1 esimhlanganise no-$OP2

Primary and secondary templates can also be linked together with 'siphinde' (and) and an 'adjoining' operator to create a chain of operators, for expressions like $x^2 + 2x +$ … and so on. The 'adjoining' operator's verb is created through a function, called adjoining_verb(), that changes the suffix of the verb to an '-e'.

(3) <primary template>, siphinde adjoining_verb() <object prefix>- <secondary template>, siphinde …

For the expression $x^2 + 2x$, this chaining template (3) generates "U-x ukhuliswe ngenkombananikuphindwa u-2, siphinde simhlanganise no-2 esimphindaphinda ngo-x." Where 'simhlanganise no-' (plus) is the operator that adjoins the primary template, 'U-x ukhuliswe ngenkombananikuphindwa u-' (power), and the secondary template '2 esimphindaphinda ngo-x'(times).

Furthermore, to handle the issue of ambiguity in operator precedence, we formulated a 'meta template' (4). The meta template uses the phrase 'futhi konke' (and all) in conjunction with the chaining template (3) to show unambiguously when an operator acts upon an entire bracket of terms.

(4) <PT>, siphinde adjoining_verb() <O1>-<ST>, ..., futhi konke <verb> <O2>-<RHS>.

Where, PT=primary template, O1=object prefix 1, ST=secondary template, O2=object prefix 2, RHS=right hand side of the expression.

Therefore expressions, such as $(x + y)^2$, can be generated with the meta template, to show the exponent operates on all the terms in the bracket: "U-x simhlanganisa no-y, futhi konke ukhuliswe ngenkombananikuphindwa u-3"

3.3 Design and Implementation of the System

The verbaliser is a Python program that runs from the terminal; it takes in an XML file containing a CMML snippet; the system processes the expression; and then outputs an isiZulu text description in a new file.

The system was designed to process mathematics formatted in CMML, because it has tagged operators, which make the expression's components much more explicit for the NLG system. Instead of focusing on the representation of mathematical formulae like MathML, CMML focuses on the semantics and meaning of expressions. The difference between the two formats are shown in Figure 1.

(1)
```
<apply>
  <divide/>
    <apply>
      <minus/>
        <ci>x</ci>
        <ci>y</ci>
    </apply>
    <cn>2</cn>
</apply>
```

(2)
```
<mfrac>
  <mrow>
    <mi>x</mi>
    <mo>-</mo>
    <mn>y</mn>
  </mrow>
  <mn>2</mn>
</mfrac>
```

Figure 1: Comparing the syntax of a (1) Content MathML and a (2) MathML Snippet for the Expression (x – y)/2

In the following subsection, we will discuss the design and implementation of the system's architecture.

3.3.1 Architecture

The system's architecture is a traditional NLG pipeline (See Figure 2), adapted from (Reiter and Dale, 1997). It is designed with distinct, well-defined, and easily integrated modules. There are three stages, that perform three of the NLG tasks proposed by Dale and Reiter: Content Determination, Document Structuring and Linguistic Realisation. A module was built for the text planner, sentence planner and linguistic realiser.

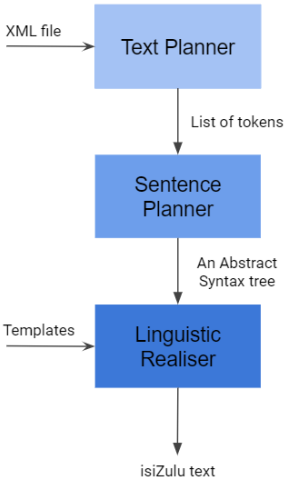

Figure 2: The Modular Pipeline Architecture, Adapted from (Reiter and Dale, 1997), for the Math Verbaliser.

The text planner module fulfills the content determination task – it receives the input CMML expression and filters out the information that will be included in the text. The file is read line by line and a list of the expression's tokens is produced. The tags containing the operands are removed, so only their value remains. The operator tags are cleaned into a more comprehensive form: '<times/>' is cleaned into 'times'. This improves ease of use and readability further down the pipeline when the templates are loaded. The apply tags are left as is. This token list is passed on to the sentence planner.

The sentence planner's job is to organize the overall structure of the intended sentence. This module takes the list of tokens as input and arranges them into an Abstract Syntax Tree (AST), (See figure 3.). The <apply> tags are used to signal the start of a new operation but are not inserted into the tree themselves.
The root of each sub-tree is the operator, the child nodes are its operands, and the leaves are the actual values of the expression (either variables or numbers). The completed AST is handed over to the linguistic realiser.

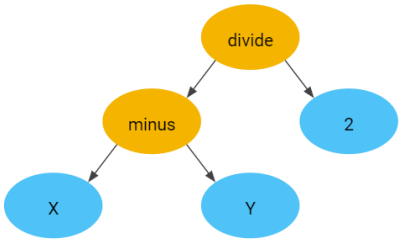

Figure 3: The AST Created After Parsing the CMML Snippet Shown in Figure 1, for the Expression (x - y)/2.

The linguistic realiser module will take the output from the sentence planner and generate a grammatically correct and complete sentence using the isiZulu templates.

To use the templates, the realiser opens a text file where the data for each template is stored and loads the file line-by-line into template objects. These objects store all the information needed to create and change a template's grammar depending on how its verb is used (as a primary, secondary or an adjoining verb). This data includes name of the CMML operator, the isiZulu verbs for the primary and secondary templates, the object prefix, and the template layout (this is important for operators that differ from the standard two slot template, like square roots). Having the template data stored externally from the realiser, allows easy adding or updating of templates.

The received AST is depth-first traversed, fetching a template via the name of the operator and filling it in for each sub-tree. The realiser evaluates the AST in Figure 3 using the meta template as "U-x simsusa ku-y, futhi konke simhlukanisa ka-2".

4 EVALUATION

This project focuses on whether the resulting system can produce understandable text for users, therefore the generated text must be evaluated by human beings. The end users of the proposed system are isiZulu speakers in the visually impaired community, however, since this evaluation focuses on the understandability of the text rather than the rendered audio, people without impairments were viable participants.

Subsection 4.1 and 4.2 discuss the method of data collection for the evaluation, and the methods of evaluating the textual descriptions.

4.1 Collection of CMML Expressions

Mathematical expressions were collected to be translated by the verbaliser and used in the evaluation. A Perl script was created to parse through any given text file and extract all CMML expressions it finds (regardless of the operations). Then the extracted CMML expressions are filtered out based on whether they contain only the chosen operators. Expressions that contain any operators that weren't chosen were discarded.
The MathML manual[1] was downloaded as a text file and the script was run on it. Approximately 50 CMML expressions that contained only the chosen operators were found. A total of 10 expressions were selected to be translated into isiZulu text using the NLG system, for the evaluation. This selection was based on their complexity; five simple expressions (containing two or less operators) and five complex expressions (containing 3 or more operators).

4.2 Methodology

The system's text was evaluated via a Google Form questionnaire. To participate, a person had to be fluent in isiZulu and have completed at least first-year mathematics. Experience in higher-level mathematics was a criterion, to ensure they had knowledge of all the operations. We recruited participants via social media posts and a survey call posted to MAM1000 students at the

[1]The MathML manual: https://www.w3.org/TR/MathML/mathml.pdf

University of Cape Town (UCT). To encourage participation, anyone who completed the questionnaire stood a chance to win a R250 raffle.
As human beings were involved, we required an ethics clearance to proceed with the evaluation, which was granted by the UCT CS Ethics Committee. The evaluation keeps the participants anonymous, as no personal or identifying information was required with the submission.

The evaluation was conducted as follows. The participants were given a link to the Google Form, where they could fill out a questionnaire. Before the questionnaire begins, participants must read and sign an informed consent form, and disclose their level of proficiency in isiZulu (First, second or third language).
The questionnaire consists of two sections per mathematical description, with a total of 10 descriptions.
Section 1 is a rating-based question, where the participants read the generated description of an expression, and then rate the understandability of the text on a Likert scale. A rating is selected from strongly disagree, disagree, unsure, agree or strongly agree. This section is where a participant's perceived understandability is measured. A small comment section is optionally available for those who would like to elaborate on their rating.
Section 2 is a task-based question, where the participants type out the corresponding formula of the description into the Google Form. The inputted formulae will be compared to the expected formula and declared as either an exact match or a mismatch. This evaluation technique assesses how accurately a reader can understand and reproduce an expression's formula based on its textual description alone, and has been used by (Mazzei et al., 2019; Mondal and Jawahar, 2019). The reason for this, is to check whether their perceived understandability corroborates with their actual understandability.
This section also aims to reveal any problems with specific operators, either issues of ambiguity, semantics, or grammar.
The following notation (see Table 3.) was provided on each page for when the participants must type out the description's formulae. The participants were also instructed to use parenthesis to show precedence between operators.

Table 3: The Notation Provided During the Evaluation, for Typing Out Mathematical Expressions.

Operator	Notation
Divide	/
Power	^
Square root	sqrt()
Times	*
Plus	+
Minus	-
Equals	=
Integral	int()

We used an attention check question halfway through the questionnaire, as a measure of the data's quality. This can potentially pick up on participants who are skim reading and not paying attention to the questions.

To analyse the responses, the ratings and exact matches from each question are added up to get a total score for each participant. Then we can determine the average rating and number of exact matches between participants. The same will be done for the individual descriptions, to see how specific templates performed. Lastly, we want to test if the complexity of a description influences whether a participant can understand its formula correctly. To test this, we will perform a Fischer exact test on the total number of matches/non-matches in each category (simple and complex), with the null hypothesis H0: There is no difference in understandability of the formulae between the simple and complex categories of the mathematical expressions.

5 RESULTS

In this section we present the findings from the human evaluation. Five isiZulu speakers with different levels of language proficiency (first, second and third language speakers) were recruited to join the questionnaire, which resulted in a total of 50 responses over 10 descriptions (the data in found in Appendix A.). We display the performance of the individual participants and the performance of the descriptions themselves.

Table 4: Compares Each Participant's Most Common Rating and the Total Number of Exact Matches Between Their Input Formula and the Expected Formula, and the Overall Average Rating and Matches between Participants.

Participant	IsiZulu Proficiency	Most Common Rating	Total Exact Matches (out of 10)
1	Second language	Unsure	3
2	third language	Agree	4
3	Second language	Agree	6
4	Second language	Strongly agree	5
5	First language	Strongly agree	7
Average		**Agree, Strongly Agree**	**5**
Standard Deviation			**1.58**

Table 4 displays the results of each participant, comparing their most common rating and their total number of exact matches out of the 10 input formulae in the questionnaire. This gives an indication of their overall understanding of the descriptions. The green cells indicate that participants 2, 3, 4, 5 mostly perceived the descriptions as understandable (i.e. they have a common rating of either 'agree' or 'strongly agree'), while the yellow cell indicates that participant 1 mostly felt unsure about the understandability of the descriptions. Participant 5, who is a first language (L1) isiZulu speaker, received the highest score: a common rating of 'strongly agree' and 7 exact matches.

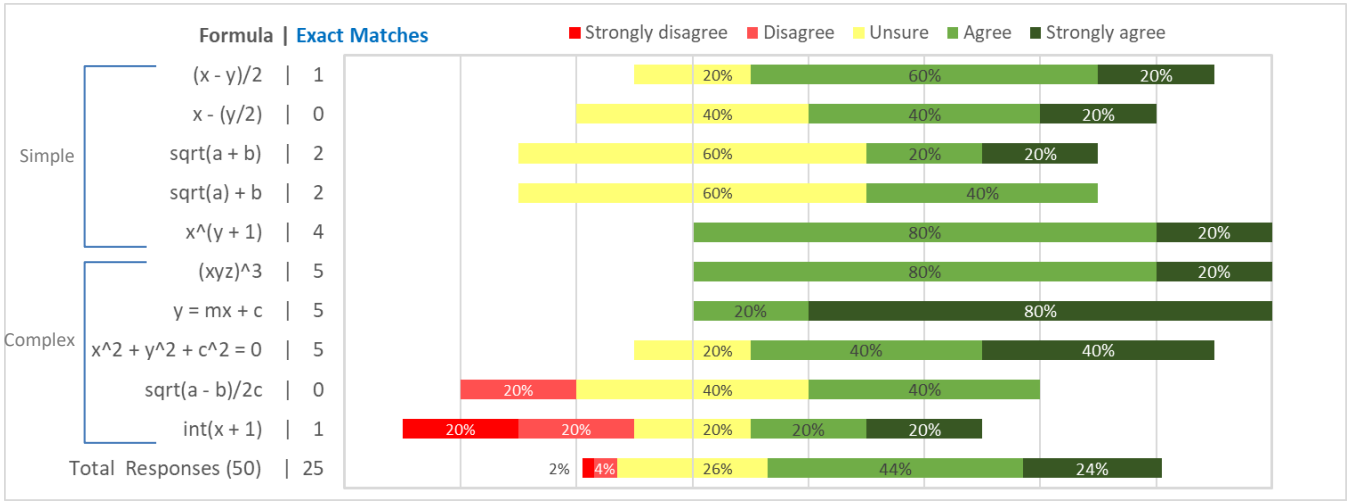

Figure 4: A Stacked Chart Comparing the Proportion of Ratings and the Total Exact Matches (out of 5) of Each Formula, Grouped by Simple or Complex; and Displays the Total Percentage of Each Rating and the Total Exact Matches for the 50 Responses.

Table 4 also indicates that the average number of exact matches per participant is 5 (± 1.58).

Figure 4 compares the performance of the descriptions, by displaying each description's proportion of ratings and total number of exact matches (shown to the right of a formula) from the five participants. To visualise the overall perceived understanding of the descriptions, the last graph of Figure 4 shows the proportion of each rating gathered from all 50 responses. If we interpret both the ratings 'agree' and 'strongly agree' as understandable, then 68% of all responses were rated as understandable. Of all the participants' responses, 26% were unsure of a description's understandability, and only 6% of all responses disagreed or strongly disagreed on a description's understandability. The total number of exact matches across all descriptions is 25 out of 50 (50%).

The expressions that performed the best included those with power, equals, multiplication, plus and divide operators.
The descriptions for the complex expressions y=mx+c and (xyz)^3 were rated as understandable by 100% of the responses, which coincides with all five participants typing out the formulae correctly. The simple description for x^(y+1) was also rated as understandable 100% of the time and received 4 exact matches. There was one more expression that did well, the complex expression x^2+y^2+z^2=0; it yielded exact matches from all five participants and was rated as understandable by 80% of responses.

The descriptions that did not perform as well, were those that contained square root, integral and minus operators. The description for the expression (x-y)/2 was interpreted as (y-x)/2 (reversed order of operands for the minus sign) by three of the five participants but was typed out correctly by one participant. The same issue with the minus operator occurred with the expression x-(y/2), it was understood as (y/2)-x by four out of five participants and received zero exact matches.

Descriptions containing a square root received mixed ratings and few exact matches. The expression sqrt(a-b)/2c could not be answered correctly by any of the participants. The L1 speaker was the only participant able to distinguish between sqrt(a)+b and sqrt(a+b). One participant, an L2 speaker, commented that they were confused by the wording 'Umaziphinde', in the square root description.
There was a single description containing an integral, int(x+1), which was answered correctly only by the L1 speaker. This description received the most mixed ratings, with a different rating from each of the five participants.

In Figure 4, simple descriptions yielded either understandable or unsure ratings, while the complex expressions had a much wider range of ratings but mostly received understandable ratings. The five simpler expressions received 9 exact matches, while the five complex expressions received 16 exact matches, from all five participants. The Fisher exact test statistic value was calculated to be 0.0887 and therefore, we cannot refute the null hypothesis at $p<0.05$. This tells us that there is no significant relationship between the complexity of an expression and its understandability.

6 DISCUSSION

The results suggest that on average a participant perceived the descriptions to be understandable, and 68% of all ratings showed that the descriptions were understandable. Based on these results from the participants' and the descriptions' performance, we can conclude that our NLG system can generate text that is generally perceived as understandable by isiZulu speakers. This implies that our realisation method of using templates with embedded functions to handle affixes, is a viable solution for building an isiZulu MathML-to-Text system.

There were two types of issues uncovered by the formula input section of the evaluation; suspected problems with terminology

for the square root and integral templates; and issues with the order of operands for the minus template.

Asking participants to type out the expression's formula was a way to determine if a participant's perceived understandability matches with their actual understandability. Although the participants perceived the descriptions as understandable, the results show that on average a participant was only able to accurately replicate the formulae 50% of the time. A good example of why this check was necessary is seen in Figure 4; 80% of participants agreed that the description for (x-y)/2 was understandable; however, three participants wrote down the same incorrect formula. Since there was a consistent error across these three participants, it's likely that the error came from the semantics of the template, and not a misunderstanding from the participants.

There were some unexpected discoveries from the results. It was expected that there would be a relationship between the complexity of expressions and the number of exact matches it gets. Surprisingly, the hypothesis test proved that complexity of a formula is not a factor in the understandability of it. However, this fact strengthens our suspicions that the misunderstandings of descriptions containing square root and integral operators, are caused by issues with semantics rather than the complexity of the description.

There are a few limitations to consider when discussing these results. The biggest limitation we faced was the small number of participants, and thus limiting the generalisability of our results. The second limitation being that four of the five participants were not L1 isiZulu speakers. Not surprisingly, the participant who spoke isiZulu as a L1 received the highest score. Therefore, it is plausible that participants having second and third language proficiency in isiZulu may have contributed to a poorer overall understanding in the results. Other limitations included possible fatigue or inattention experienced by participants. This questionnaire could be quite long and tiring, causing participants to hurry through the questionnaire and respond regardless of their true abilities, leading to possibly skewed and invalid responses. The attention check question was included as a safeguard against this, however, this method it is not fool proof. Lastly, too many participants answered with an 'unsure' rating. This rating made up 26% of responses, which is vague and can be interpreted very differently by each respondent. Although there was a comment section, no one offered further information on why they found that the descriptions were not understandable, or for what reason they felt unsure. This leaves us uncertain of which part of a description was misleading.

A recommendation for going forward is that the templates for the minus, square root, and integral operators require improvements; we consider that during the corpus creation stage, collecting more output samples from a wider range of isiZulu speakers and then basing our templates off the most popular responses, could be a more comprehensive approach to building templates, as shown by (Ferres and Sepúlveda, 2011) in their template-based system.

7 CONCLUSIONS AND FUTURE WORK

This study aimed to identify how we can build a MathML-to-Text system that produces text that is perceived as understandable by isiZulu speakers. We approached this problem with an NLG pipeline that uses templates, with a few word-level functions on top of them, as a realisation method to construct natural language sentences. Based on a human evaluation of the generated text, we conclude that our realisation method produced text that was averagely perceived as understandable among isiZulu speakers; and thus, is an appropriate technique for building a MathML-to-Text system.

While these results clearly illustrate a general perceived understandability, they also revealed that respondents could translate the description into its correct formulae only 50% of the time. This was due to a consistent problem among participants with the minus, square root and integral templates, and not a misunderstanding on the participants' behalf.

We also established that there is no relationship between the complexity of a formula and its understandability, which further demonstrations that this gap in perceived understanding and actual understanding is likely a fault in the semantics of the templates, rather than the complexity of the formulae. Hence, going forward these three templates will require improvements to their terminology.

In future work, we hope to extend the collection of templates to include more CMML operators and involve several L1 isiZulu speakers during the process of collecting corpus data. Other potential work includes implementing sentence variation, for more natural, non-repetitive output.

REFERENCES

S. Grover, A., Van Huyssteen, G., and Pretorius, M. (2011). The South African human language technology audit. Language Resources & Evaluation, 45, pp 271-288.

A. Gatt and E. Krahmer: Survey of the State of the Art in Natural Language Generation: Core tasks, applications, and evaluation. J. Artif. Intell. Res. 61: pp 65-170, 2018

E. Reiter and R. Dale: Building Applied Natural Language Generation Systems, Cambridge University Press, 1997

L. Ferres and J.F. Sepúlveda: Improving accessibility to mathematical formulas: the Wikipedia math accessor. Proceedings of the International Cross-Disciplinary Conference on Web Accessibility, pp. 1-9. 2011.

A. Mazzei, M. Monticone, C. Bernareggi: Using NLG for speech synthesis of mathematical sentences. Proceedings of the 12th International Conference on Natural Language Generation, pp 463–472. 2019.

C. M. Keet and L. Khumalo. "Grammar rules for the isiZulu complex verb". In: Southern African Linguistics and Applied Language Studies 35.2, pp. 183–200, 2017

Z. Mahlaza and C. Maria Keet: A classification of grammar-infused templates for ontology and model verbalisation, In: Metadata and Semantic Research - 13th International Conference, MTSR, Rome, Italy 2019.

T. V. Raman. Audio System for Technical Readings. PhD thesis, Computer Science, Cornell University, 1994.

N. Soiffer. Mathplayer: web-based math accessibility. Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility, Baltimore, MD, USA, pp 204–205. ACM Press, 2005.

D. Ahmetovic, T. Armano, C. Bernareggi, et al: Axessibility: A LaTeX package for mathematical formulae accessibility in pdf documents. In Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility, pp 352–354, 2018.

A. Mondal and C. Jawahar: Textual Description for Mathematical Equations, Centre for Visual Information Technology, International Institute of Information Technology, Hyderabad, India, Published in ICDAR, 2019

Appendix A. - Supplementary Information

Table 5: Raw Data Captured from the Online Questionnaire, which was Conducted with Five Users.

	User 1	User 2	User 3	User 4	User 5
Informed Consent	Yes	Yes	Yes	Yes	Yes
	16/09/2020	16/09/2020	19/09/2020	12/01/2000	25/09/2020
	Z.P	LM	S. K.	SY	LS
Proficiency	Second language	Third language	Second language	Second language	First/home language
D1	Agree	Agree	Agree	Unsure	Strongly Agree
	(Y-X)/2	(y-x)/2	(X-y)/2	(y-(x/2))	(Y-X)/2
					Yes
D2	Unsure	Agree	Unsure	Strongly Agree	Agree
	Y/2-x	(y/2)-x	(X-y)/2	(y/2)-x	X(y/2 -x)
					Yes
D3	Unsure	Unsure	Unsure	Strongly Agree	Agree
	a+b	a+b	Sqrt(a+b)	(sqrt(a))+b	Sqrt(a+b)
	My Zulu is bad,my home language is isiXhosa and I thought will understand fully (confused by mazphinde)				Yes
D4	Unsure	Unsure	Agree	Unsure	Agree
	2a+b	a+b	Sqrt(a+b)	(sqrt(a))+b	Sqrt(a)+b
					Yes
D5	Agree	Agree	Agree	Agree	Strongly Agree
	x^y-1	x^(y+1)	xy*(y+1)	x^(y+1)	X^y+1
					Yes
D6	Agree	Agree	Agree	Agree	Strongly Agree
	(X x Y x Z)^3	(x*y*z) ^3	(X*y*z)*3^2	(xyz)^3	(X*Y*Z)^3
					Yes
Att. Check	Unsure	Agree	Strongly Agree	Strongly Agree	Strongly Disagree
	x/y	x/y	X/y	x/y	X/Y
					Yes
D7	Strongly Agree	Agree	Strongly Agree	Strongly Agree	Strongly Agree
	Y = mx + c	y=m*x+c	Y=Mx+c	y=mx+c	Y=mx+c
					Yes
D8	Agree	Agree	Unsure	Strongly Agree	Strongly Agree
	x^2 + y^2+ z^2=0	x^2+y^2+z^2=0	x^2+2y+2z=0	(x^2)+(y^2)+(z^2)=0	(X^2)+(y^2)+(z^2)=0
					Yes
D9	Unsure	Disagree	Agree	Agree	Unsure
	(b-2a)/2a	B-a	Sqrt(a-b)-(2-a)	(b-sqrt(a))/2a	Sqrt(b-a)=2a
					Yes
D10	Strongly Disagree	Unsure	Disagree	Strongly Agree	Agree
	2+1	Sqrt(x) +1	(X+1)/2	1/(x+1)	Int(x+1)