Building a Spoken Dialogue System for Supporting Blind People in Accessing Mathematical Expressions

Pier Felice Balestrucci[1], Luca Anselma[1], Cristian Bernareggi[2] and Alessandro Mazzei[1]

[1]*Dipartimento di Informatica, Università degli Studi di Torino, Italy*
[2]*Laboratorio Polin - Dipartimento di Matematica, Università degli Studi di Torino, Italy*

Abstract
English. Mathematical expressions are complex hierarchical structures of symbols that are usually accessed by visual inspection. These expressions are seldom rendered with natural language since users are not usually required to read them aloud. People with a visual impairment generally use LaTeX with *screen readers* to acquire mathematical expressions. However, LaTeX can be verbose, slow to listen, and difficult to learn. This work proposes a way to make mathematical expressions easier to be accessed by people with disabilities by exploiting their hierarchical structures. We describe and evaluate a dialogue system to vocally navigate mathematical expressions in English. In contrast with standard screen readers, the vocal interaction allows people to query the system about sub-parts of the expressions.

Italiano. Le espressioni matematiche sono complesse strutture gerarchiche di simboli generalmente esplorate visivamente. Queste espressioni raramente vengono rappresentate tramite linguaggio naturale, perché agli utenti di solito non è richiesto di leggerle ad alta voce. Le persone con disabilità visiva di solito usano il LaTeX con uno *screen reader* per ascoltare le espressioni matematiche. Tuttavia, il LaTeX può essere verboso, lento da ascoltare e difficile da imparare. Questo lavoro propone una alternativa per rendere più accessibili le espressioni matematiche alle persone con disabilità sfruttando la loro struttura gerarchica. Descriviamo ed valutiamo quindi, un sistema di dialogo per navigare vocalmente le espressioni matematiche in inglese. A differenza dei normali screen reader, l'interazione vocale consente alle persone anche di interrogare il sistema sugli elementi che compongono le espressioni.

Keywords
dialogue system, natural language processing, visually impaired people

1. Introduction

In the last few years, improvements in speech technologies have enabled the use of Natural Language Processing in various application domains. Among these, the field of assistive technologies stands out as one with the most potential. Specifically, spoken dialogue systems (SDSs) can enhance the quality of life for numerous users with special needs.

In this paper we study the task of accessing mathematics for blind people. Traditionally, people with a visual impairment, in order to acquire mathematics, use LaTeX with *screen readers*, that are dedicated text-to-speech software. However, LaTeX is verbose, slow to listen, and not well known in young people. Indeed, LaTeX is designed as a typographical language, that is a technical language to provide typographical details rather than a compact description of a mathematical content. Moreover, mathematical expressions can range over multiple dimensions (e.g. fractions are arranged on two levels) and the linear encoding of LaTeX can produce a very long sequence with distant relations among symbols closely connected.

To tackle these issues, in this paper we propose a SDS based on the main idea to use standard natural language to interact with mathematical contents. For instance by using a SDS people with a visual impairment do not need to use a keyboard and a mouse to use their device. They can activate vocally the SDS to interact directly. Furthermore, this approach can also address other disabilities, including motor impairments. The SDS uses *mathematical sentences*, that are natural language sentences containing the semantics of a mathematical expression [1, 2]. There are many advantages in using mathematical sentences with respect to LaTeX. First, mathematical sentences are shorter. Second, they are not semantically ambiguous as some LaTeX expressions (e.g. $f(x)$ could also stand for $f \cdot (x)$). Third, they do not require knowledge of LaTeX, so that they can be used by a wider class of users (e.g. children). In particular, the SDS converts the LaTeX encoding of the mathematical expression in a semantic language, called Content Math Markup Language (CMML)[1]. From CMML, English mathematical sentences are generated according to good practices for spoken mathematics [3], through a standard Natural Language Generation (NLG)

CLiC-it 2023: 9th Italian Conference on Computational Linguistics, Nov 30 — Dec 02, 2023, Venice, Italy
pierfelice.balestrucci@unito.it (P. F. Balestrucci); luca.anselma@unito.it (L. Anselma); cristian.bernareggi@gmail.com (C. Bernareggi); alessandro.mazzei@unito.it (A. Mazzei)

[1]https://www.w3.org/TR/MathML3/chapter4.html

architecture, i.e. a sentence planner and a realizer.

The interaction design of the SDS is quite straightforward: the SDS pronounces the mathematical sentences and the user can interrupt it to "navigate" the expression (e.g. ask for repetition of its parts). So, the dialogue manager component coordinates the recognition of a repetition command with the generation of (a subpart of) the mathematical sentence.

We performed a user-based evaluation on the effectiveness and the usability of the SDS. The system implementation and evaluation were conducted with the involvement of visually impaired experts. The results show that the developed system has a good impact both on the comprehension of mathematical expressions and on the user experience, constituting a promising approach for helping people with visual impairments.

The paper is structured as follows: In Section 2, we report related work. In Section 3, we describe the main components of the SDS. In Section 4, we describe a human-based evaluation of the SDS and in Section 5, we closes the paper with some consideration and future work. Finally, Section 6 discusses some limitations of our SDS.

2. Related Work

2.1. Access to mathematics by people with visual impariments

Many different solutions have been investigated to enable people with visual impariments to access mathematical expressions. They can be divided up into two main categories: systems to read mathematics and systems to type and simplify maths expressions. The former category includes applications to read LaTeX in PDF files [4, 5, 6, 7], maths in web pages through speech rendering of MathML [8] or MathJax [9, 10], to read source LaTeX documents [11, 12] and maths in R Markdown [13]. These solutions propose and evaluate reading models based on sequential reading or hierarchical reading of maths expressions based on keyboard interaction. To the best of our knowledge, to date no studies have investigated a dialogue system to facilitate reading and exploration on maths expressions through speech input and output.

The latter category includes specialized applications that are designed to facilitate students with visual impairments to simplify expressions in LaTeX format [14] or in a multimodal work environment specifically designed for inclusive classes [15, 16] and to work with bi-dimensional mathematical procedures as arithmetic operations [17].

2.2. Speech-to-text solutions for entering maths expressions

This section introduces the solutions which have been investigated to write mathematics through speech input. TalkMaths [18] [19] is a prototype application which translates a limited set of arithmetic, algebraic and trigonometric expressions from spoken English into LaTeX or MathML. It adopts Dragon Naturally Speaking[2] (DNS) as speech recognition system. The translation rules are defined only for English and the recognition implements a dictation model based on pauses, which slow down the dictation process [20].

Mathifier [21] is an open source software module which converts a subset of mathematical expressions from English into LaTeX. It combines a dictionary, a language model and an acoustic model to recognize mathematical English utterances. It is based on Sphinx-4 [22] to recognize speech. This project has not been maintained and updated regularly.

CamMath [20] is a proof of concept prototype application designed to prove the advantages of continuous speech over discrete utterance of mathematical expressions in English.

Metroplex MathTalk[3] is a commercial application that provides speech input of arithmetic, algebra, calculus and statistics in English.

EquatIO[4] enables dictation of simple maths expressions in English in MS Word and in GSuite applications.

Even though these applications have been designed to enable speech recognition of mathematical expressions, none of them has addressed the needs of people with visual impairments by combining speech input and speech output.

3. Spoken Dialogue System

In this section we describe the main components of the SDS. As most rule-based SDSs [23], the information flow follows a path initiated by the user, who starts the interaction with a request to read a specific mathematical expression (Fig. 1). The SDS pronounces the mathematical sentence. The user listens to the produced sentence, and possibly interrupts the SDS asking for clarification. At this point the SDS answers to the request and the dialogue goes on. In Fig. 1 we report the architecture of the SDS, which is based on three main components, that are the response generator (described in Section 3.1), the language understanding and the dialogue management (described both in Section 3.2).

[2]https://www.nuance.com/it
[3]www.metroplexvoice.com/
[4]https://www.texthelp.com/en-us/products/equatio/

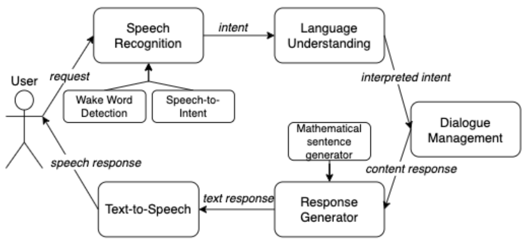

Figure 1: Dialogue System Architecture

3.1. Response Generator and Text-to-Speech

The generation side of the SDS follows the traditional NLG architecture composed of document planning, sentence planning and realizer [24]. In particular, for the mathematical sentence generation, we follow the pipeline described in Mazzei et al. [1, 2]. Note that, in contrast with Mazzei et al. [1], we model the grammar of the mathematical sentence for English rather than Italian for this novel SDS. Moreover, we propose an SDS, whereas in Mazzei et al. [1] we proposed a pure NLG system, where the interaction was limited to listening to the generated sentence.

The document planning consists of an encoding of the mathematical expression from LaTeX into a semantically unambiguous format, i.e. CMML, through the LatexML tool [25], specifying some heuristics defined by the specific mathematical sub-domain (e.g. algebra).

Let us consider the following formula as an example: "$A \times B = \{(x, y) \mid x \in A, y \in B\}$" and its LaTeX representation: "A \times B = { (x, y) \mid x \in A, y \in B}". The mathematical formula will be converted in the following CMML format:

```
<apply>
<eq/>
    <apply>
    <cartesianproduct/>
        <ci>A</ci>
        <ci>B</ci>
    </apply>
    <apply>
    <conditional-set/>
        <apply>
        <pair/>
            <ci>x</ci>
            <ci>y</ci>
        </apply>
        <apply>
        <and/>
            <apply>
            <in/>
                <ci>x</ci>
                <ci>A</ci>
            </apply>
            <apply>
            <in/>
                <ci>y</ci>
                <ci>B</ci>
            </apply>
        </apply>
    </apply>
</apply>
```

The CMML representation of the above formula is unambiguous: for each operator (empty tag–e.g. <eq/>), there is an opening and closing tag (i.e. <apply>...</apply>). Within these tags, we can find nested parts of the mathematical sentence such as other operators and the variables that compose the arguments of each operator.

For implementing the sentence planning, we model the syntactic structure of the English mathematical expressions by using as reference the linguistic constructions presented in Chang [3], a standard reference point for spoken mathematics in assistive technologies. With this linguistic reference for English, following the approach of Mazzei et al. [1], we divide the mathematical operators in categories characterized by the same arity, and for each one we define a syntactic template. For instance, the operators in Table 1 (e.g. $+$) are generally modeled with declarative sentence (e.g. $a\ plus\ b$), while elementary functions (e.g. sin) are modeled with a noun phrase (e.g. $sin\ of\ x$).

Symbol	Operator	English Form
$+$	plus	plus
$-$	minus	minus
$/$	divide	over
$*$	times	times
$[...]^{[..]}$	power	to
$\backslash$	settdiff	minus
$\times$	cartesianproduct	cross
$\cap$	intersect	the intersection of
$\cup$	union	the union of ... and ...

Table 1
Algebraic, Arithmetic and Set Operators

Note that within a category there are still different ways to compose a sentence. Taking as an example the minus operator, it can appear as: (1) $minus\ 5$, (2) $a\ minus\ b$ (a noun phrase and a declarative phrase, respectively). The difference between these two sentences depends on how the operator is used. In (1) $minus$ is used to define a negative value, in (2) $minus$ is used as the difference between two variables. From a realisation point, we need to distinguish two different forms within the category of algebraic, arithmetic and set operators: (1) unary form (e.g.

minus 5): a so-called adjective phrase must be defined where the operator (e.g. *minus*) works as an adjective; (2) binary form (e.g. *a minus b*): a declarative structure must be defined where the operator (e.g. *minus*) works as the parent node of the expression.

The realisation phase uses SimpleNLG [26], which is a Java library for morphological realization and linearization in English. Note that we added special symbols into SimpleNLG lexicon to produce both parentheses and pauses. Finally, we control the pronunciation, volume and pitch by using the Speech Synthesis Markup Language (SSML)[5]. As the last step, the mathematical sentence is pronounced through a Text-to-Speech technology.

We experiment with two different vocal synthesizers: the commercial AWS Polly[6] and the open source eSpeak[7] (both support SSML). AWS Polly is based on an advanced deep learning technology and has a human-kind voice, which is clear and highly user-adjustable, whereas eSpeak is based on a so-called formant synthesis method and produces a robotic voice. It is worth noting that eSpeak is a very familiar voice for visually impaired people. The users can choose which vocalizer to use.

3.2. Speech Recognition, Language Understanding and Dialogue Management

The speech recognition module in Fig. 1 is composed of two different sub modules: Wake-up word detection and Speech-to-Intent. A wake-up word [27] is a keyword that triggers the speech-to-intent module. The classification is binary and happens in real time. For this task we used Porcupine (v.2.1) [28] which has good results in comparison to other commercial systems. A Speech-to-Intent system is able to recognize a user's intent in a very specific context. The system works on a small vocabulary of terms and classifies each user's request. For this task we used Rhino (v.2.1) [29] which allowed us to detect in real time the user's vocal commands for our mathematical context.

The intent is translated into a request which is interpreted by the language understanding module with regular expressions, which matches the intent produced by the speech recognition with a domain specific speech act. These speech acts are: (1) repetition, (2) query and (3) resuming. The repetition intent lets the user ask the repetition (from a subpart) of the mathematical sentence (e.g. "Repeat from the first integer"). In this case the system searches within the content response what the user wants to be repeated. The query intent lets the user ask the system to repeat a specific part of the mathematical sentence (e.g. "What is the limit of the second integer?"). In this case, the specific operator in the user's request will be searched. If matched, the system produces an answer with the requested part of the operator. Finally, the resuming intent lets the user resume the interaction after an interruption (e.g. "Go on").

The main algorithm of this module is represented below:

```
dialogue():
1.  Say "I'm starting to say the sentence"
2.  Activate the Wake-up Word Detection
    system
3.  For each word within the sentence:
4.      Say the word
5.      If the Wake-up Word Detection
        system has detected "Hey stop":
6.          Say "Ok, I'm listening to you"
7.          Activate the Request
            Recognition system
8.          Fulfill the request
9.  Say "I've finished reading this
    sentence, but I'm still here for you"
10. Activate the Request Recognition
    system
```

In lines 1 and 9 the system announces to the user the start and the end of the expression. After the activation of the wake-up word detection system, for each word the system will pronounce it. If the user says the wake-up word, the system will stop and put itself into a listening phase waiting for a request. Completed the expression, the System can continue to answer user's requests.
The "fulfill the request" method in line 8 works as follows:

```
fulfill_request(intent, request):
1.  Parsify the request
2.  If the intent is "Query":
3.      Search for an operator in the
        request
4.      Match the request with the
        operator arguments
5.      If the match is successful:
6.          Answer the question
7.          Otherwise, utter an
            apology message
8.  If the intent is "RepetitionFrom":
9.      Search within the sentence
        uttered what the user
        wants to hear repeated
10.     If the search is successful:
11.         Repeat from that point
12.         Otherwise, utter an
            apology message
13. If the intent is "Resume":
14.     Resume the synthesis of the
        mathematical expression
```

[5]https://www.w3.org/TR/speech-synthesis11/
[6]https://aws.amazon.com/it/polly/
[7]http://espeak.sourceforge.net/

If the intent has been interpreted as repetition, the system will search within the content response what the user wants to be repeated. If the user's intent has been interpreted as query, the specific operator in the user's request will be searched. If matched, the system will answer the requested part of the operator. The dialogue management module is in charge of making decisions. If the interpreted request does not correspond to any of the supported possibilities, it asks the user to repeat the request. Otherwise, the dialogue management searches the content of the response for the user and sends it to the response generator module.

4. Evaluation

To evaluate the SDS, we conducted two experiments involving visually impaired users with the approval of the University's ethical committee. For the first experimentation we tested the effectiveness and solidity of the English generation system in a similar way as Mazzei et al. [1] and for the second experiment the effectiveness and usability of the SDS. In Experiment 1 we recruited two blind native Italian speakers proficient in English with an excellent maths knowledge that participated freely and without compensation. We retrieved from a calculus textbook [30] 10 mathematical expressions of different length and difficulty (Table 2 and Table 3), and we represented them in CMML. The difficulty of a formula is related to the number of parentheses and the number of nodes in its CMML representation. Then, we used SDS to synthesize the expressions. Using the same experimental setting of Mazzei et al. [1], we obtained 25 audios (10 easy ones generated using different synthesizers, i.e. eSpeak and Polly, and 15 difficult ones generated with different strategies for generating pauses, e.g. for parentheses). We uploaded the audio files to Youtube and provided them to the users along with a questionnaire[8] (on Google Form, because it is accessible) containing profiling questions and the request to write down the expressions in the audio files in an unambiguous notation.

We evaluated the written expressions with two metrics: Exact Match (EM) and SPICE [31]. EM is 1 if the original CMML and the one obtained by the user are the same, and 0 otherwise. SPICE is obtained by calculating the F-score of the overlapping between the original CMML tree and the one obtained from the user. The overlapping is measured by decomposing the CMML trees in typed elementary substructures, which are operands, operators and their relations. Experiment 1 results (Table 4) show that the generation system seems to be effective since the users obtained a good understanding of the expressions.

LaTeX Formula	Nodes
$\sqrt[n]{x} = x^{1/n}$	10
$x > b \Longrightarrow \lvert f(x)\rvert < M$	10
$g^{-1}(y) = f^{-1}\left((y-b)/a\right)$	13
$\int_b^c a \,\mathrm{d}x = a(c-b)$	14
$A \times B = \{(x,y) \mid x \in A, y \in B\}$	15

Table 2
Easy expressions - Few parenthesises and nodes - Experiment 1

LaTeX Formula	Nodes
$\lim \left(1+\frac{1}{n}\right)^n = e$	10
$\int \frac{1}{\sqrt{m^2-x^2}}\mathrm{d}x = \arcsin\frac{x}{m} + c$	20
$y = f(a) + \frac{f(b)-f(a)}{b-a}(x-a)$	21
$\sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x-x_0)^k$	28
$\lim_{x\to x_0}\left\{\frac{f(x)-f(x_0)}{x-x_0} - f'(x_0)\right\} = 0$	31

Table 3
Difficult expressions - More parenthesises and nodes - Experiment 1

User	Metrics	Tot. (25)	Easy (10)	Difficult (15)
1	EM	0.92	1.00	0.87
	SPICE	0.98	1.00	0.97
2	EM	1.00	1.00	1.00
	SPICE	1.00	1.00	1.00
avg	EM	0.96	1.00	0.93
	SPICE	0.99	1.00	0.99

Table 4
EM and SPICE - Experiment 1

In Experiment 2 we recruited five blind native Italian speakers that declared a good maths knowledge and proficiency in English; however, one of the users dropped out because of their low competency. The users participated freely and without compensation. In Experiment 2 the users connected via Google Meet to a client running the SDS. This modality has been decided on the basis of COVID-19 restrictions still in force at that time. A

[8]https://docs.google.com/forms/d/e/1FAIpQLSc93-1NAIWy_SXQaTJH2tAnHKg9PHSysurOSzf3xbSvtKG7Ig/formResponse?pli=1

facilitator established the Google Meet connection, presented the experimental protocol (see the instructions presented to the users in Fig. 2), and observed the user interactions while remaining neutral. After a short time (about 30 minutes) when they could practice with the SDS, the users were presented with 3 easy and 3 difficult expressions chosen among the 10 expressions of Experiment 1 (Table 5 and Table 6). The SDS used Polly for the speech synthesis. The users interacted autonomously with the SDS and they could interrupt the system and ask questions. Finally, the users were asked to write down the expressions questionnaire similarly to Experiment 1.

```
First of all, thank you for taking part in this
experimentation.
During today’s test you will have to interact
with a voice assistant who will pronounce six
mathematical phrases to you. Your job is to write
them in any understandable formalism,
even in English, and send me your answers by
the Google Meet chat or by email.
The voice assistant is able to receive
various commands when interrupted by saying “Hey stop!”.
I recommend you spell these words well and use a high
volume of voice. The commands are:
 1. Repeat from anywhere in the sentence.
 For example, consider the expression “a + b”:
 if “Say again from plus” is said, the
 assistant will repeat “+ b”. The keyword
 is “Say again from” and you can
 repeat a variable or an operator.
 2. Specify an operator.  For example consider
 the expression “a + b + c” you can ask
 “What is the first sum?” or “What is the
 left argument of second sum”. The keyword is
 “What is” and you can ask for information about
 a variable or an operator. Regarding the
 operator you can ask how you have already
 heard to repeat the argument in its case,
 specifying whether left or right, lower or upper.
 3. In the event that you accidentally interrupted
 the assistant or do not remember the question you
 wanted to ask him, just say “Go on”.
If you try to stop the assistant without success,
please repeat “Hey stop!” several times. From the moment
it stops you have about 1 minute maximum to make the
request. Let’s take a few examples now,
if you don’t have any questions!
```

Figure 2: Instructions for testers - Experiment 2

LaTeX Formula	Nodes
$\sqrt[n]{x} = x^{1/n}$	10
$g^{-1}(y) = f^{-1}\left((y-b)/a\right)$	13
$\int_b^c a \, \mathrm{d}x = a(c-b)$	14

Table 5
Easy expressions - Few parenthesises and nodes - Experiment 2

LaTeX Formula	Nodes
$\int \frac{1}{\sqrt{m^2-x^2}} \mathrm{d}x = \arcsin\frac{x}{m} + c$	20
$y = f(a) + \frac{f(b)-f(a)}{b-a}(x-a)$	21
$\lim_{x\to x_0} \left\{ \frac{f(x)-f(x_0)}{x-x_0} - f'(x_0) \right\} = 0$	31

Table 6
Difficult expressions - More parenthesises and nodes - Experiment 2

As in Experiment 1, we used EM and SPICE as evaluation measures. Moreover, the users also compiled the User Experience Questionnaire (UEQ) [32] to evaluate their experience in terms of attractiveness, perspicuity, efficiency, dependability, stimulation and novelty[9].

The scores for Experiment 2 are worse than the ones of Experiment 1 both on EM and on SPICE (cf. Table 7). This could be explained by the complexity of the setting, because users had to learn how to use a new tool in a short time, whereas the users in Experiment 1 were familiar with the linear fruition of a Youtube audio. We observed that the performance of the users improved over the time of the experiment, which can be due to an acquired familiarity of the tool.

In Table 8 we present the results of the UEQ along different attributes. For each attribute we report the score of the attribute on a scale between -3 and $+3$ and compare it with a benchmark provided by UEQ [32] that includes a dataset with 468 products evaluated by $21,175$ users. It is worth noting that the SDS scored high over stimulation, novelty and attractiveness and fair over perspicuity, efficiency and dependability. The scores over efficiency

[9]https://www.ueq-online.org/

User	Metrics	Tot. (6)	Easy (3)	Difficult (3)
1	EM	0.50	0.67	0.33
	SPICE	0.86	0.89	0.87
2	EM	1.00	1.00	1.00
	SPICE	1.00	1.00	1.00
3	EM	0.33	0.33	0.67
	SPICE	0.66	0.98	0.82
4	EM	0.33	0.00	0.67
	SPICE	0.80	0.95	0.89
avg	EM	0.54	0.50	0.67
	SPICE	0.83	0.95	0.89

Table 7
EM and SPICE - Experiment 2

Attribute	Score	Comparison to benchmark
Attractiveness	1.83	between 75% and 90%
Perspicuity	1.28	between 50% and 75%
Efficiency	1.33	between 50% and 75%
Dependability	1.33	between 50% and 75%
Stimulation	1.75	above 90%
Novelty	1.92	above 90%

Table 8
UEQ Benchmark

and novelty are consistent with the hypothesis that users would benefit from a longer training time to become proficient with this new tool, that they however deem stimulating and attractive. These preliminary experiments seem to be promising, nevertheless it would be beneficial to enlarge the pool of users. However, it is known in accessibility studies [33] that involving visually impaired people in experiments is significantly hard and several studies tend to engage only sighted people.

5. Conclusion

In this paper we described a SDS designed for allowing visually impaired people to access mathematical expressions. In Experiment 1 we focused on the understanding of the mathematical sentences generator for English (i.e. using EM and SPICE measures), replicating the good results obtained for Italian in Mazzei et al. [1, 2]. In Experiment 2, we tested the complete SDS allowing user to ask for repetition. With respect to expressions understanding, the results of this experimentation are less encouraging than Experiment 1, but we speculate that this is a consequence of the complexity of the experimental setting due to the necessity of online interaction. However, the UEQ showed that the users really appreciated the interaction with the SDS. In the future we want to improve the SDS by adding new intents. Moreover, we want to define a new SDS designed for diagrams and other visual structures, creating accurate descriptions and making them navigable.

6. Limitations

The SDS developed in this paper has two main limitations. The design interaction is limited to repetition request concerning a subpart of the expression. A better interaction could consider the possibility to ask for mathematical clarification on the role of a subpart (e.g. “what is x?”). The evaluation has two aspects that could be improved: 1. the limited number of testers, and 2. no native English speakers participated.

References

[1] A. Mazzei, M. Monticone, C. Bernareggi, Using NLG for speech synthesis of mathematical sentences, in: Proceedings of the 12th International Conference on Natural Language Generation, Association for Computational Linguistics, Tokyo, Japan, 2019, pp. 463–472. URL: https://aclanthology.org/W19-8658. doi:10.18653/v1/W19-8658.

[2] A. Mazzei, M. Monticone, C. Bernareggi, Evaluating speech synthesis on mathematical sentences, in: Proceedings of the Sixth Italian Conference on Computational Linguistics, Bari, Italy, November 13-15, 2019, 2019, pp. 1–7. URL: http://ceur-ws.org/Vol-2481/paper46.pdf.

[3] L. A. Chang, Handbook for Spoken Mathematics, The Regent of the University of California, 1983.

[4] T. Armano, A. Capietto, S. Coriasco, N. Murru, A. Ruighi, E. Taranto, An automatized method based on latex for the realization of accessible pdf documents containing formulae, in: International Conference on Computers Helping People with Special Needs, Springer, 2018, pp. 583–589.

[5] D. Ahmetovic, T. Armano, C. Bernareggi, A. Capietto, S. Coriasco, D. Boris, K. Alexandr, N. Murru, et al., Automatic tagging of formulae in pdf documents and assistive technologies for visually impaired people: the latex package axessibility 3.0, in: ICCHP 2020 17th International Conference on Computers Helping People with Special Needs, volume 1, ICCHP, 2020, pp. 69–73.

[6] D. Ahmetovic, T. Armano, C. Bernareggi, M. Berra, A. Capietto, S. Coriasco, N. Murru, A. Ruighi, E. Taranto, Axessibility: A latex package for mathematical formulae accessibility in pdf documents, in: Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility, 2018, pp. 352–354.

[7] D. Ahmetovic, T. Armano, C. Bernareggi, A. Capietto, S. Coriasco, N. Murru, et al., Axessibility 2.0: creating tagged pdf documents with accessible formulae, Ars Texnica (2019) 138–145.

[8] N. Soiffer, Browser-independent accessible math, in: Proceedings of the 12th International Web for All Conference, 2015, pp. 1–3.

[9] D. Cervone, P. Krautzberger, V. Sorge, Employing semantic analysis for enhanced accessibility features in mathjax, in: 2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC), IEEE, 2016, pp. 1129–1134.

[10] V. Sorge, C. Chen, T. Raman, D. Tseng, Towards making mathematics a first class citizen in general screen readers, in: Proceedings of the 11th Web for All Conference, 2014, pp. 1–10.

[11] R. Chauhan, I. Murray, R. Koul, Audio rendering

of mathematical expressions for blind students: a comparative study between mathml and latex, in: 2019 IEEE International Conference on Engineering, Technology and Education (TALE), IEEE, 2019, pp. 1–5.

[12] A. Bansal, M. Balakrishnan, V. Sorge, Comprehensive accessibility of equations by visually impaired, ACM SIGACCESS Access. Comput. 126 (2020) 1. URL: https://doi.org/10.1145/3386280.3386281. doi:10.1145/3386280.3386281.

[13] J. Seo, S. McCurry, A. Team, Latex is not easy: Creating accessible scientific documents with r markdown, Journal on Technology and Persons with Disabilities 7 (2019) 157–171.

[14] S. Arooj, S. Zulfiqar, M. Qasim Hunain, S. Shahid, A. Karim, Web-alap: A web-based latex editor for blind individuals, in: The 22nd International ACM SIGACCESS Conference on Computers and Accessibility, volume 28, 2020, pp. 1–6.

[15] V. Sorge, Supporting visual impaired learners in editing mathematics, in: Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility, 2016, pp. 323–324.

[16] C. Bernareggi, Non-sequential mathematical notations in the lambda system, in: ICCHP, Springer, 2010, pp. 389–395.

[17] A. Gerino, N. Alabastro, C. Bernareggi, D. Ahmetovic, S. Mascetti, Mathmelodies: inclusive design of a didactic game to practice mathematics, in: International Conference on Computers for Handicapped Persons, Springer, 2014, pp. 564–571.

[18] A. Wigmore, G. Hunter, E. Pflügel, J. Denholm-Price, V. Binelli, Using automatic speech recognition to dictate mathematical expressions: The development of the "talkmaths" application at kingston university., Journal of Computers in Mathematics and Science Teaching 28 (2009) 177–189.

[19] A. M. Wigmore, E. Pflugel, G. J. Hunter, J. Denholm-Price, M. Colbert, Talkmaths better! evaluating and improving an intelligent interface for creating and editing mathematical text, in: 6th International Conference on Intelligent Environments, IEEE, 2010, pp. 307–310.

[20] C. Elliott, J. Bilmes, Computer based mathematics using continuous speech recognition, Vocal Interaction in Assistive Technologies, Games and More (2007).

[21] S. N. Batlouni, H. S. Karaki, F. A. Zaraket, F. N. Karameh, Mathifier—speech recognition of math equations, in: 18th International Conference on Electronics, Circuits, and Systems, IEEE, 2011, pp. 301–304.

[22] W. Walker, P. Lamere, P. Kwok, B. Raj, R. Singh, E. Gouvea, P. Wolf, J. Woelfel, Sphinx-4: A flexible open source framework for speech recognition, 2004.

[23] K. Jokinen, M. McTear, Spoken Dialogue Systems, Synthesis lectures on human language technologies, Morgan & Claypool Publishers, 2010. URL: https://books.google.it/books?id=ualwulnD020C.

[24] E. Reiter, R. Dale, Building Natural Language Generation Systems, Natural Language Processing, Cambridge University Press, 2000. URL: http://prp.contentdirections.com/mr/cupress.jsp/doi=10.2277/052102451X. doi:DOI:10.2277/052102451X.

[25] B. Miller, Latexml: A LaTeX to XML converter, https://math.nist.gov/~BMiller/LaTeXML, 2007.

[26] A. Gatt, E. Reiter, SimpleNLG: A realisation engine for practical applications, in: Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009), 2009, pp. 90–93.

[27] Y. Wang, Wake word detection and its applications, Johns Hopkins University, 2021.

[28] Picovoice, Benchmarking a Wake Word Detection Engine, https://picovoice.ai/blog/benchmarking-a-wake-word-detection-engine/, 2018.

[29] Picovoice, Picovoice Console — Rhino Speech-to-Intent Engine, https://picovoice.ai/docs/quick-start/console-rhino/, 2018.

[30] L. Pandolfi, ANALISI MATEMATICA 1, Dipartimento di Scienze Matematiche "Giuseppe Luigi Lagrange", Politecnico di Torino, 2013.

[31] P. Anderson, B. Fernando, M. Johnson, S. Gould, SPICE: semantic propositional image caption evaluation, CoRR, abs/1607.08822, 2016.

[32] M. Schrepp, User experience questionnaire handbook. all you need to know to apply the ueq successfully in your project, https://www.ueq-online.org/, 2015.

[33] E. Brulé, B. J. Tomlinson, O. Metatla, C. Jouffrais, M. Serrano, Review of quantitative empirical evaluations of technology for people with visual impairments, in: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, 2020, pp. 1–14.