

UNIVERSIDAD NACIONAL DE CORDOBA
FACULTAD DE MATEMÁTICA, ASTRONOMÍA Y FÍSICA

TRABAJO FINAL DE GRADO

**Una propuesta de accesibilidad en
expresiones matemáticas para la
comunidad de ciegos y disminuidos
visuales**

Autor:

Luis THUR

Supervisoras:

Paula ESTRELLA

Laura ALONSO ALEMANY

23 de Febrero de 2016



Contents

1	Introducción	1
1.1	Descripción del problema	1
1.2	Motivación	2
1.3	Alcance de la tesis	4
1.4	Estructura de la tesis	4
2	Análisis de tipos de entrada	6
2.1	MathML	6
2.1.1	¿Qué es MathML?	6
2.1.2	PMathML	7
2.1.3	CMathML	7
2.1.4	¿Por qué CMathML?	8
2.2	\LaTeX	10
2.2.1	¿Qué es \LaTeX ?	10
2.2.2	Análisis de \LaTeX y MathML	11
3	Arquitectura de la solución	13
3.1	Entrada y salida	13
3.2	A favor de una solución modular	14
3.3	SnuggleTeX y Cliente SnuggleTeX	16
3.4	Pre-procesador	20
3.5	Generador de Lenguaje	22
3.5.1	Gramática	25
4	Construcción y evaluación de la gramática	27
4.1	Reglas descriptas en la gramática	27
4.2	¿El resultado es la representación más usual?: Evaluación	28
4.3	Descripción del procedimiento	29
4.3.1	Recolección de datos	30
4.3.2	Pre-pocesamiento de datos	31
4.3.3	Corpus building	33
4.3.4	Constructor de evaluadores	35
4.3.5	Recolección de resultados	40
A	Appendix Title Here	45

Chapter 1

Introducción

1.1 Descripción del problema

La accesibilidad a la matemática es un importante tópico para que la educación sea cada vez más inclusiva. Aún así, como lo es de importante puede que lo sea de complejo. En este trabajo se intenta resolver unos de los problemas planteados por alumnos de carreras con matemáticas incluidas en su programa de la Facultad de Matemática Astronomía, Física y Computación, de la UNC.

Uno de los mayores problemas de acceso a la matemática que afectan a la comunidad de estudiantes y científicos ciegos es la lectura de expresiones matemáticas. El problema es que estas poseen un componente fuertemente visual que para una persona ciega se vuelve completamente inaccesible.

Éste componente visual es la imagen que muestra una expresión matemática. Afortunadamente esa imagen muchas veces está presente en un archivo pdf y es generada usando \LaTeX , si es un recurso que se puede obtener y manipular, se puede pensar en la accesibilidad.

La propuesta de solución consiste principalmente en la lectura de fórmulas matemáticas a partir de expresiones matemáticas escritas en lenguaje \LaTeX . Los sujetos afectados por el problema han encontrado una alternativa, utilizan un lector de pantallas (de ahora en adelante ScreenReader) que es un sistema TTS (Text-To-Speech) y su función principal es de la lectura de toda cadena de caracteres presente en la pantalla con

el fin de mejorar la accesibilidad. El inconveniente es que las expresiones matemáticas escritas en \LaTeX son leídas literalmente como se muestran y resulta ser una tarea difícil de comprender para el sujeto que lo esté escuchando.

Por ejemplo, la siguiente fórmula matemática: $\frac{1}{2} + \frac{1}{2} = 1$ que en \LaTeX se escribe:

`\frac{1}{2}+\frac{1}{2}=1`

el ScreenReader lee *barra invertida frac uno dos mas barra invertida frac uno dos igual uno* mientras que nosotros naturalmente leemos *Un medio más un medio es igual a uno*.

Es por eso que este trabajo busca mejorar la accesibilidad de la matemática desarrollando una aplicación que, en un paso previo a la acción del ScreenReader, describa en lenguaje natural las expresiones matemáticas para que luego sean leídas naturalmente.

1.2 Motivación

Durante el Censo Nacional de Población, Hogares y Viviendas del año 2010 también denominado Censo del Bicentenario se arrojaron resultados relativos a la Población con dificultad o limitación permanente (PDLP). En aquel año, Argentina contaba con una población de 39.671.131 de habitantes de los cuales 5.114.190 habitantes presentaban alguna dificultad o limitación permanente, que representa un 12,9 sobre el total de la población.

El censo declara en su artículo que se cataloga por discapacitada a toda persona que padezca una alteración funcional permanente o prolongada, física o mental, que en relación a su edad y medio social implique desventajas considerables para su integración familiar, social, educacional o laboral.

Sexo	Población en viviendas particulares	Población con dificultad o limitación permanente	Prevalencia de la dificultad o limitación permanente
Total	39.671.131	5.114.190	% 12,9
Varones	19.276.217	2.263.175	11,7
Mujeres	20.394.914	2.851.015	14,0

Fuente: INDEC. Censo Nacional de Población, Hogares y Viviendas 2010.

FIGURE 1.1: Población con dificultad o limitación permanente y prevalencia de la dificultad o limitación permanente, por sexo. Año 2010

Concentrándonos en la parte de la población que presenta alguna dificultad visual, el censo obtuvo como resultado que la distribución porcentual con tal dificultad representa el 59,9 de la población con dificultades, mientras que el resto de las dificultades (Motora superior, Motora inferior, Cognitiva y Auditiva) se reparten en proporciones casi iguales.

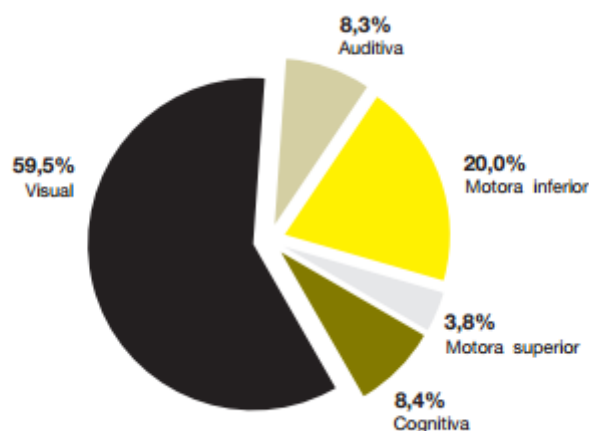


FIGURE 1.2: Distribución porcentual de la población con una sola dificultad o limitación permanente por tipo de dificultad. Total del país. Año 2010

Por lo tanto, la cantidad de personas con discapacidades visuales fue de 3.063.399 en el año 2010. Por otra parte, el 90,2 por ciento de la población concurre o concurrió a un establecimiento educativo común, no de educación especial, al que solo fue o va el 9,8 de los encuestados. El 26,1 por ciento de los encuestados usa computadora, y es la población de entre 6 y 29 años la que más lo usa.

La inclusión a la educación es realmente un importante tópico para poder incrementar estos números, aportar a la inclusión ayudará a la población a insertarse en los distintos niveles educativos.

1.3 Alcance de la tesis

Se desea construir una prueba de concepto (PoC) que intente resolver el problema planteado. Esta PoC busca entender diferentes entradas:

- una expresión matemática escrita en **LaTeX**
- una expresión matemática escrita en **MathML**
- **archivo** que contenga expresiones matemáticas escritas en **LaTeX**.

Dado cualquiera de estas entradas, la PoC hará traducción al lenguaje natural en español de las expresiones matemáticas y será devuelto en texto. Además, la PoC asumirá la existencia de un convertor de LaTeX a MathML (adelante en el documento se explicará la necesidad de éste componente), y no generará TTS. Ésta PoC busca que funcione modularmente entre las distintas entradas y el sistema TTS preferido por el usuario.

1.4 Estructura de la tesis

El siguiente documento de tesis se divide en cinco secciones principales. En la sección 1 (y actual) se mencionará una introducción al documento de tesis, le brindará al lector una idea introductoria del problema que la tesis busca resolver, a su vez, busca dar un contexto social al problema y la importancia de su solución. A su vez, esta sección busca definir qué alcance tendrá el documento y su proyecto asociado.

En la sección 2 - *Análisis de tipos de entrada* - se muestra un análisis de cómo iniciar a encarar el problema que se quiere resolver. Mostrando qué tipos de entradas son las que manipula los sujetos involucrados. También, esta sección, busca desarrollar con el lector los términos y definiciones que serán utilizados a lo largo del documento, conocer en detalle qué ofrece y qué no ofrece cada entrada. Y finalmente, mostrar

los argumentos que llevan a la decisión de cómo iniciar el proceso de desarrollo de la solución.

En la sección 3, - *Arquitectura de la solución* -, busca mostrarle al lector cuáles partes están involucradas y que conjuntamente forman a la solución final. Esta sección define los distintos alcances, objetivos, acciones, comportamiento de cada componente. A su vez, quiere desplegar la manera en la que estos componentes se comunican.

En la sección 4, llamada *Construcción y evaluación de la gramática*, despliega al lector los mecanismos empleados para la construcción de éste verbalizador que busca mejorar la accesibilidad de la matemática y a su vez, muestra al lector 3 maneras de evaluar el verbalizador. El objetivo de ésta sección es poder mostrar cómo el verbalizador puede mejorar para que se asemeje cada vez mas a cómo nosotros mismos dictaríamos una expresión matemática a una persona que no puede verla.

Finalmente si bien los anexos no son una sección en particular aquí está presente toda la evidencia de la que el resto del documento cita y se basa. Intenta que el lector tenga más contexto si desea averiguar un poco más acerca de lo que está leyendo.

Chapter 2

Análisis de tipos de entrada

2.1 MathML

2.1.1 ¿Qué es MathML?

MathML (Mathematical Markup Language) es un dialecto XML para describir notaciones matemáticas que busca capturar tanto la estructura como el contenido. Su objetivo es integrar las fórmulas matemáticas en las páginas de Internet y otros documentos. Forma parte de HTML5 y de un estándar ISO ISO/IEC DIS 40314 desde el año 2015.

Una de las ventajas de MathML es que es un estándar de representación de fórmulas matemáticas para la World Wide Web (W3) y los principales y más usados navegadores de Internet tienen integrado de manera nativa el soporte para éste tipo de contenido haciéndolo una aplicación accesible.

MathML consiste de un número de tags XML que pueden ser usados para el marcado de una ecuación en terminos de su representación y su semántica. Ésto último es sumamente importante ya que MathML busca capturar el significado detrás de la ecuación además de concentrarse en cómo ésta será formateada y mostrada en pantalla.

MathML busca facilitar el uso y el re uso de contenido científico en la Web. La versatilidad de éste lenguaje de marcado nos permite aprovechar su notación para displays visuales de alta calidad y su contenido para aplicaciones más semánticas como software científico o sintetizadores de voz.

Como se mencionó MathML codifica tanto la estructura como el contenido de las expresiones matemáticas. A través del marcado de presentación (Presentation MathML o PMathML) se pueden desplegar expresiones matemáticas, y mediante el marcado de contenido (Content MathML o CMathML) se puede representar el significado de las expresiones matemáticas.

2.1.2 PMathML

El **marcado de presentación** tiene como objetivo principal describir la estructura de una fórmula matemática. Los elementos de presentación corresponden a los **constructores** de la notación matemática tradicional, en la última versión de MathML se incluyen 37 tags que pueden ser usados para representar una fórmula matemática. Los tags más usados son `<mi>`, `<mo>`, `<mrow>` para indicar una *variables*, *operadores*, *expresiones anidadas*, respectivamente.

2.1.3 CMathML

El **marcado de contenido** tiene como objetivo principal describir la semántica de una fórmula matemática. Provee una lista mucho más amplia de tags para ser usados con respecto a PMathML, sumando un total de 129. Entre los tags más usados tenemos `<apply>`, `<ci>`, `<cn>`. El primero de ellos se utiliza para hacer explícita la aplicación de un operador a sus childrens (tags hijos), dejando en claro cual es el scope de aplicación. El segundo tag se utiliza para describir variables y el tercero de ellos se utiliza para describir los números. CMathML contiene un tag en específico para cada operación matemática, lo que lo hace más específico que PMathML ya que PMathML contiene un tag solo para algunas operaciones (como la suma ó multiplicación).

En la siguiente figura se puede observar cómo se describe en PMathML y CMathML para la siguiente fórmula matemática: $x + \frac{a}{b}$

```

<apply>
  <plus/>
  <ci>x</ci>
  <apply>
    <divide/>
    <ci>a</ci>
    <ci>b</ci>
  </apply>
</apply>

```

FIGURE 2.1:
CMathML

```

<mrow>
  <mi>x</mi>
  <mo>+</mo>
  <mfrac>
    <mi>a</mi>
    <mi>b</mi>
  </mfrac>
</mrow>

```

FIGURE 2.2:
PMathML

2.1.4 ¿Por qué CMathML?

Hay ciertas expresiones matemáticas que pueden tener distintas interpretaciones, esto la hace una expresión ambigua, ya que muchas veces depende del contexto su correcta interpretación. Supongamos la siguiente formula matemática:

$$f^{-1}$$

Que puede interpretarse como la función inversa de f o, también como, la variable f elevado a -1 . Para ello veamos como es su marcado tanto en PMathML como en CMathML. Para PMathML tenemos el siguiente XML:

```

<math xmlns="http://www.w3.org/1998/Math/MathML">
  <msup>
    <mi>f</mi>
    <mn>-1</mn>
  </msup>
</math>

```

Podemos ver que, dado que PMathML intenta solo capturar su presentación, su XML asociado sigue siendo ambiguo. Ya que la etiqueta `<msup>` solo señala que -1 es el superíndice de f . Lo que lo vuelve ambiguo tal como se escribe su expresión matemática.

En cambio para CMathML, ya que ésta intenta capturar su semántica, podemos ver que tenemos dos posibles XML asociados a esta fórmula matemática.

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <apply>
    <inverse/>
    <ci type="function">f</ci>
  </apply>
</math>
```

y también tenemos

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <apply>
    <power/>
    <ci type="function">f</ci>
    <cn>-1</cn>
  </apply>
</math>
```

Podemos ver que para CMathML queda claro de qué interpretación queremos asociarle al XML. Ya que en el primer caso tenemos el tag `<inverse>` aplicado a la variable f que habla de la inversa de f . Y en el segundo tenemos el tag `<power>` que denota la exponencia de -1 aplicado a f .

Otra ventaja que tiene CMathML sobre PMathML es que existen herramientas para hacer el pasaje de CMathML a PMathML, pero debido a la ambigüedad de PMathML no existe una herramienta que lo haga en la otra dirección.

Esta es la razón principal que CMathML es el adecuado para la PoC que acompaña este trabajo final, es el lenguaje de marcado que más se acerca a la lectura en lenguaje natural.

2.2 L^AT_EX

2.2.1 ¿Qué es L^AT_EX?

Según Wikipedia, L^AT_EX es un sistema de composición de textos, orientado a la creación de documentos escritos que presenten una alta calidad tipográfica. Por sus características y posibilidades, es usado de forma especialmente intensa en la generación de artículos y libros científicos que incluyen, entre otros elementos, expresiones matemáticas.

Lo cierto es que L^AT_EX posee a su disposición muchos paquetes para ser importados que hacen más fácil la escritura de expresiones matemáticas. Cualquier expresión matemática puede ser escrita usando L^AT_EX.

Latex fue diseñado específicamente para ser escrito por humanos por eso éste sistema es el más usado para la generación de archivos científicos y, por lo tanto, muchas expresiones matemáticas son escritas con L^AT_EX, lo que hace que L^AT_EX sea el lenguaje más usado para escribir éste tipo de expresiones. Si bien MathML es usado por distintos softwares científicos y navegadores web, no es realmente el lenguaje más usado para escribir expresiones matemáticas desde el lado del usuario, como lo es L^AT_EX.

Esta es la razón por la que L^AT_EX es considerado como un tipo de entrada válido para la PoC.

A modo ilustrativo, veamos cómo es en sintaxis L^AT_EX la fórmula $x + \frac{a}{b}$:

```
$x + \frac{a}{b}$
```

Podemos notar que, al igual que CMathML, captura la estructura y el contenido de la expresión. Sabiendo que `\frac` es la manera de escribir una fracción y con `+` podemos escribir la suma, podemos leer con facilidad la expresión matemática mencionada y darle el respectivo significado. Cabe mencionar, que hay una relación entre CMathML y L^AT_EX y será explicado más adelante en este documento.

L^AT_EX además de ser usado ampliamente por científicos para escribir sus publicaciones y documentos, también es usado para describir algunas expresiones matemáticas en Internet. Tal es así el caso de Wikipedia, que para cada fórmula matemática que

tenga presente en algún artículo en línea posee dentro de el atributo *alt* del tag `` su descripción en lenguaje \LaTeX .

Por ejemplo, veamos en el artículo que explica las ecuaciones de segundo grado

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

mientras que su código HTML tenemos

```

```

2.2.2 Análisis de \LaTeX MathML

Si bien la relación entre Latex y CMathML es muy sensible y fina, existe. Comencemos diferenciado éstas tecnologías desde un punto de vista de usabilidad y comprensión, así también desde tipo de entrada.

- **Usabilidad:** Latex es el lenguaje más usado para escribir documentos y artículos científicos, hay tutoriales y documentación con ejemplos para aprender a escribir usando éste lenguaje. Latex funciona desde el lado del usuario (*user-side*), mientras que MathML es la aplicación más usada para renderizar expresiones matemáticas en distintos softwares, y funciona desde el lado del servidor (*server-side*). En muchos artículos se menciona que MathML es la representación que las computadoras entienden de las expresiones que conocemos.
- **Compresión desde el usuario:** Una expresión matemática, tanto escrita con Latex o MathML, pueden ser leídas con el mismo grado de dificultad. Si la expresión es mas avanzada, la lectura desde un humano puede requerir un poco más de atención. Pero a la hora de la escritura, Latex es más amigable que MathML.
- **Tipo de entrada:** MathML saca ventaja sobre Latex de ésto ya que, como se mencionó, MathML es un dialecto XML que nos permite renderizar cualquier expresión matemática y consigo trae todas las ventajas que tiene una aplicación

XML. El parser es un componente estándar, por lo tanto no es necesario crear un parser específico para cada versión de lenguaje XML. Esto posibilita el empleo de cualquiera de los analizadores disponibles. De esta manera se evitan bugs y se acelera el desarrollo de aplicaciones. Pero no funciona así con Latex, debido que Latex es un lenguaje de programación y no un estándar puede ser extendido por lo tanto no hay un parser disponible para poder particionar una expresión latex para procesar partes más atómicas.

Las conclusiones hasta el momento es que Latex es el lenguaje más elegido por los usuarios y MathML es la aplicación más elegida para desarrollar el software. Es por eso que en ésta tesis se intenta hacer un ligamento entre éstas tecnologías y aprovechar las ventajas de cada uno.

Chapter 3

Arquitectura de la solución

3.1 Entrada y salida

Se asumen tres tipos de entradas que TexToES puede entender.

- **LaTeX**

Dadas las razones explicadas anteriormente, Latex es un tipo de entrada que nuestra PoC puede entender e interpretar. TexToES asume que la expresión latex está encerrada entre \$ para delimitar el inicio y el fin de ésta.

Ejemplo:

$$\$ \backslash \max (1, 2, 3, 4) = 4 \$$$

- **CMathML**

Como se mencionó, CMathML captura el contenido de la expresión es por eso que se prefiere CMathML antes que PMathML. TexToES asume que la entrada viene dada con su respectiva estructura XML definida por el estándar. Por lo tanto, deberá comenzar con la etiqueta **<math>**.

Ejemplo:

$$\langle \mathbf{math} \rangle \langle \mathbf{apply} \rangle \langle \mathbf{eq} \rangle \langle \mathbf{ci} \rangle a \langle \mathbf{ci} \rangle \langle \mathbf{ci} \rangle b \langle \mathbf{ci} \rangle \langle \mathbf{apply} \rangle \langle \mathbf{math} \rangle$$

- **Archivo que contenga expresiones latex**

TexToES también admite cualquier archivo con extensión **.txt* que dentro contenga expresiones latex. Se debe cumplir también aquí que las expresiones esten encerradas entre $\$$.

Para cada tipo de entrada obtenemos un texto como tipo de salida que es la representación en lenguaje natural español de la entrada selecta. Para visualizarlo mejor, tenemos el siguiente gráfico:



FIGURE 3.1: Flujo simple de TexToES

Los engranajes representan **TexToES** y las flechas señalan la dirección del flujo. Más adelante en este documento explicaré con más detalle cómo se compone la PoC.

3.2 A favor de una solución modular

La manera en que se encaró el desarrollo de TexToEs fue de manera tal que cada componente estuviera aislado y separado del resto. Llevando a cabo la estrategia *divide y vencerás* fue que se logró programar cada componente por separado, con una interfaz definida para que el resto de los componentes puedan comunicarse a través de ella.

Se decidió distribuir las distintas tareas identificadas en módulos para que se pueda reusar código, para poder adaptar nuevas funcionalidades y la razón más importante es poder utilizar TexToES como un único módulo a la preferencia del usuario o desarrollador.



FIGURE 3.2: Gráfico de flujo de TexToES

TexToES cuenta con cinco principales módulos los cuales son:

- Cliente SnuggleTeX: es el cliente que se encarga de comunicarse con el servidor

SnuggleTeX, el cliente implementa una sencilla interfaz para hacer y parsear el request que convierte LaTeX a MathML. Esto entra en acción si el usuario quiere verbalizar texto LaTeX.

- Pre-procesador: Es un simple módulo que pre-procesa el MathML recibido (ya sea por el usuario o el cliente) añadiendo etiquetas y elementos que ayuden al procesamiento del XML, como resultado provee un stack que será manipulado por el verbalizador.
- Verbalizador o Generador de lenguaje: Se encarga de parsear el stack que representa la fórmula matemática e ir verbalizando cada operador, constructor y constante que encuentre.
- Gramática: define la gramática utilizada por el Verbalizador.
- Score: Sistema que evalúa la transcripción obtenida. Indica métricas que muestran el grado de confianza de que tiene el usuario de que su verbalización obtenida esté bien.

La combinación de éstos módulos forman a TexToES. Cabe mencionar que cada uno de ellos es exportable, i.e. el lector puede importarlos desde su ambiente de desarrollo y los puede usar por separado utilizando su interfaz.

3.3 SnuggleTeX y Cliente SnuggleTeX

SnuggleTeX es una librería Java gratuita y open-source para convertir fragmentos de Latex a XML (usualmente XHTML + MathML). Las funcionalidades principales que SnuggleTex provee son:

- SnuggleTeX convierte de código matemático simple en LaTeX a MathML, puedes obtener tanto PMathML como CMathML.
- SnuggleTeX puede convertir el MathML resultante en imágenes (usando la librería JEuclid) y además puede convertir código MathML simple en una mezcla de XHTML y CSS.

- SnuggleTeX puede tambien enriquecer la salida PMathML que crea, adicionalmente generando CMathML y/o Maxima.

Entre otras funcionalidades.

SnuggleTeX es un componente altamente complejo pero que nuestra PoC lo usa como un componente de caja negra, es decir, no conocemos cómo está implementado solo utilizamos su interfaz y obtenemos sus resultados.

SnuggleTeX es quien se encarga de convertir el codigo LaTeX, que viene por entrada, a codigo CMathML para luego alimentar de éste al resto de los componentes para obtener la verbalización.

Dado que ésta herramienta provee un mecanismo de conversion de un tipo a otro, podemos olvidarnos de la creación de un parser en específico para LaTeX que intente traducirlo a MathML, simplemente alimentamos a SnuggleTeX con el código Latex que deseemos verbalizar y luego manipulamos su salida en CMathML. Actualmente, ST se encuentra hosteado en un servidor para proyectos académicos donde se puede consumir toda su funcionalidad.

Para poder alimentar y extraer la información de ST, se implementó un cliente HTTP que pueda comunicarse con el servidor que hostea a la herramienta. El cliente conoce dónde está ubicado éste servidor, conoce tambien, las distintas URIs de cada funcionalidad y también conoce los nombres de los campos con los que ST va a consumir nuestra petición.

Comunicarse con el servidor de ST es básicamente realizar un pedido POST al servidor indicando en el campo *input* el string LaTeX a convertir. Luego el cliente se encarga de parsear su respuesta para obtener el XML asociado, y en caso de que no esté presente devuelve un objeto **nulo**.

¿Cuáles son los casos en que puede no estar presente el XML asociado?

Para responder ésta pregunta, nos concentramos en las limitaciones de SnuggleTeX.

ST tiene un rango limitado de operadores que puede soportar, si el LaTeX que le demos de entrada posee un operador o símbolo que no esté en la siguiente lista no obtendremos un resultados que podamos verbalizar.

La lista es la siguiente:

Operadores soportados

Operador LaTeX	Content MathML	Ejemplo
+	<plus/>	<u>$x+1$</u>
-	<minus/>	<u>$-a-b$</u>
Cualquier multiplicación	<times/>	<u>$A\times 3x$</u>
Cualquier división	<divide/>	<u>$1/2/{3\div 4}$</u>
\vee	<or/>	<u>$A\vee B$</u>
\wedge	<and/>	<u>$A\wedge B$</u>
\cup	<union/>	<u>$A\cup B$</u>
\cap	<intersect/>	<u>$A\cap B$</u>
\setminus	<setdiff/>	<u>$A\setminus B\setminus C$</u>
\not	<not/>	<u>$\not A$</u>
!	<factorial/>	<u>$x!!$</u>
Cualquier operador de relación	Ver abajo	<u>$1\leq x < y$</u>

FIGURE 3.3: Operadores soportados por SnuggleTeX

Para los operadores lógicos tenemos:

Operadores de relación soportados

LaTeX	CMathML	Ejemplo
=	<code><eq/></code>	$x=1$
<code>\not=</code>	<code><neq/></code>	$x\not=a$
<	<code><lt/></code>	$a<b$
<code>\not<</code>	<code><not>...<lt/>...</not></code>	$a\not<b$
>	<code><gt/></code>	$a>b$
<code>\not></code>	<code><not>...<gt/>...</not></code>	$a\not>b$
<code>\leq</code>	<code><leq/></code>	$x\leq 1$
<code>\not\leq</code>	<code><not>...<leq/>...</not></code>	$x\not\leq 1$
<code>\geq</code>	<code><geq/></code>	$x\geq 1$
<code>\not\geq</code>	<code><not>...<geq/>...</not></code>	$x\not\geq 1$
<code>\equiv</code>	<code><equivalent/></code>	$a\equiv b$
<code>\not\equiv</code>	<code><not>...<equivalent/>...</not></code>	$a\not\equiv b$
<code>\approx</code>	<code><approx/></code>	$x\approx 1$
<code>\not\approx</code>	<code><not>...<approx/>...</not></code>	$x\not\approx 1$
<code>\mid</code>	<code><factorof/></code>	$a\mid b$
<code>\not\mid</code>	<code><not>...<factorof/>...</not></code>	$a\not\mid b$
<code>\in</code>	<code><in/></code>	$a\in A$
<code>\notin</code>	<code><notin/></code>	$a\notin A$

FIGURE 3.4: Operadores lógicos soportados por SnuggleTeX

También ST puede parsear y dar significado en MathML para funciones pre-definidas.

```

\sin \cos \tan \sec \cot \sinh \cosh \lcm \ln \log
\tanh \sech \csch \coth \arcsin \arccos \max \Re
\arctan \arcsec \arccsc \arccot \arcsinh \min \det
\arccosh \arctanh \arcsech \arccsch \gcd \Im \exp \arccoth

```

Quizás, en algún trabajo futuro, se pueda completar la implementación de SnuggleTeX para aquellos operadores que no estén listados aquí y así agrandar la esfera de acceso de ésta herramienta.

Finalmente, TexToES utiliza ST cuando se alimente nuestra PoC con un string que represente LaTeX. Para los casos en donde directamente dispongamos de una entrada en XML CMathML, ST no estará involucrado en éste escenario. Luego con el XML CMathML, ya sea provisto por el usuario u obtenido con el cliente de SnuggleTeX, entra en acción el siguiente módulo en el flujo.

3.4 Pre-procesador

El Pre-Procesador (o pre-processor) es el módulo más simple en TexToES pero no menos complejo. Recibe como entrada una cadena de caracteres (string) que represente un XML del tipo CMathML:

Ejemplo: `<math><apply><eq/><ci>a</ci><ci>b</ci></apply></math>`

Para devolver una pila (stack) con la estructura del árbol XML en una lista utilizando el método de búsqueda **depth-first**.

Éste último método mencionado es un algoritmo muy conocido en el área de Computación que permite recorrer todos los nodos de un árbol de manera ordenada, pero no uniforme. Su funcionamiento consiste en ir expandiendo todos y cada uno de los nodos que va localizando en un camino concreto. Cuando ya no quedan más nodos que visitar en dicho camino, regresa haciendo backtracking, de modo que repite el mismo proceso con cada uno de los hermanos del nodo ya procesado.

Se ha utilizado la librería estándar de Python para parsear XML (*xml.etree*).

A medida que se recorre el árbol utilizando DF, los elementos del mismo se van apilando utilizando las siguiente reglas de apilación:

- El primer nodo, identificado como "**math**", se descarta.
- Por cada nodo etiquetado como "**apply**" se apila en el stack el string "**apply**", que es quien indica el inicio de la aplicación de un operador.
- Por cada nodo etiquetado como "**end**", se apila en el stack el string "**end-apply**" similar al anterior, solo que con ésto podemos indicar el fin de la aplicación de la operación.
- Cada vez que se encuentre un nodo "**ci**" o "**cn**" se apila directamente el contenido de su nodo hijo. (Recordemos que ci y cn representan números y variables respectivamente).
- Cualquier otro caso se apila el nodo sin ninguna alteración.

Cómo habrán notado, la etiqueta "**end**" no está presente en el string de entrada. Esto es así porque el pre-processor añade esta etiqueta para poder identificar cuando finaliza la aplicación de una operación. Esta etiqueta custom es una etiqueta dummy su única función es identificar el final de la etiqueta "apply".

Para ello hay un algoritmo simple implementado en éste módulo, podemos describir su funcionamiento en la siguiente oración:

- Por cada `</apply>` presente en el texto, añadiré a su izquierda lo siguiente `<end></end>`. De manera que finalmente quedará reemplazado por `<end></end></apply>`.

Al finalizar, se entrega una lista (llamada stack) con el contenido del árbol recorrido en DF.

La salida del pre-processor para el ejemplo presentado es la siguiente:

Salida: ['apply', 'eq', 'A', 'B', 'end-apply']

3.5 Generador de Lenguaje

Éste módulo se encarga de verbalizar el stack que recibe desde el pre-processor. Si bien parece simple leyendo esa descripción pero es el módulo más complejo. El modulo provee una manera de generar texto basado en templates (llamado en procesamiento de lenguaje natural como template-based generators).

Procederé a explicar el funcionamiento de éste módulo mediante un ejemplo práctico. Partamos de la siguiente expresión matemática:

$$3x - 2 = 0$$

El cual se puede representar con el siguiente CMathML:

```
<math>
  <apply>
    <eq/>
    <apply>
      <minus/>
      <apply>
        <times/>
        <cn>3</cn>
        <ci>x</ci>
      </apply>
      <cn>2</cn>
    </apply>
    <cn>0</cn>
  </apply>
</math>
```

Y el stack que obtenemos gracias al pre-processor (y entrada de éste módulo) es el siguiente:

```
['apply', 'eq', 'apply', 'minus', 'apply', 'times', '3', 'X', 'end-apply', '2', 'end-apply', '0', 'end-apply']
```


En rangos generales el mecanismo empleado por éste módulo se puede describir con las siguientes reglas:

1. ¿'apply' está presente en stack?
 - (a) si no, terminar.
 - (b) caso contrario:
 - i. i es la posición de 'apply' más a la derecha en stack.
 - ii. j es la posición de 'end-apply' próximo a la derecha de i
 - iii. creo un stack temporal llamado stack_temp que contiene el sub array desde i hasta j
 - iv. obtengo la operación
 - A. consulto en la gramática por la traducción de la operación
 - v. obtengo su traducción desde la gramática
 - vi. obtengo la aridad, basandome en la cantidad de operandos en stack_temp
 - vii. obtengo la verbalización de stack_temp
 - viii. elimino de stack los elementos desde i hasta j y coloco su verbalización
 - (c) vuelvo al primer paso

Para el caso planteado podemos ver que 'apply' está presente 3 veces en el stack, por lo que el algoritmo descripto anteriormente se aplicará 3 veces.

Primera iteración:

El elemento 'apply' más a la derecha presente en stack en ésta iteración es el que está en la posición 4. Su end-apply próximo está en la posición 8. Por lo tanto el sub stack llamado stack_temp es el siguiente

['apply', 'times', '3', 'X', 'end - apply']

Los datos que sé es que la operación en éste stack atómico es '**times**', su aridad es **2** y la gramática (que detallaré más adelante) nos expone que su traducción es **\$VAR\$ multiplicado por \$VAR\$** donde \$VAR\$ es un símbolo de la gramática que me puede exponer 1 o más transformaciones consecuentes. En éste caso, dado que la aridad es 2, cada \$VAR\$ será reemplazado por los operados (3 y X).

La verbalización consiste simplemente en parsear la traducción desde la gramática y saber colocar desatendidamente los operados en los respectivos \$VAR\$.

Si verbalización finalmente es

$$3multiplicadoporX$$

En el stack original debemos reemplazar todos los elementos stack_temp presentes en stack por la verbalización obtenida en esta iteración. Finalmente el siguiente es nuestro stack resultante de la primer iteración

$$['apply', 'eq', 'apply', 'minus', '(3multiplicadoporX)', '2', 'end - apply', '0', 'end - apply']$$

Segunda iteración:

El apply siguiente se encuentra en la posición 2 y su correspondiente end-apply en la posición 6. La operación es **minus** y su aridad es también **2**. La gramática nos dice que su traducción es **\$VAR\$ menos \$VAR\$**. Finalmente su verbalización se corresponde con:

$$((3multiplicadoporX)menos2)$$

Y el stack resultante es:

$$['apply', 'eq', '((3multiplicadoporX)menos2)', '0', 'end - apply']$$

Tercera iteración:

De manera análoga, obtenemos que el resultado final es

$$(((3multiplicadoporX)menos2)esigual0)$$

3.5.1 Gramática

Como se ha mencionado anteriormente, este sistema que busca verbalizar, o traducir, el latex al español tiene su sistema de generación de lenguaje natural implementado con plantillas, o lo que comúnmente se conoce como template-based generation systems. TexToES posee una gramática donde mapea las operaciones soportadas (descriptas por el operador, o función, en la etiqueta CMathML) con su representación en lenguaje natural. Es simplemente un archivo de configuración donde por cada clave (key) tenemos los operadores y en los valores (value) tenemos su representación en español.

La decisión por la que se llevó a cabo la implementación del template es porque se consideró que cada elemento de la etiqueta CMathML tiene una correspondencia uno a uno al español.

Por ejemplo, si estamos hablando de **<plus>** sabemos que estamos hablando de una suma aplicada a dos o más operados. Por lo tanto, en nuestra gramática encontraremos la siguiente línea.

plus = \$VAR\$ más \$VAR\$

Dónde \$VAR\$ corresponde con símbolos terminales o con otra key dentro de la gramática. Gracias a ésto último podemos describir una suma anidada de números, o simplemente la suma de dos números.

Sea \mathbb{T} todos los elementos que denoten una variable. Por ejemplo, $x, y, f, g \in \mathbb{T}$.

Sea \mathbb{OP} todos los operadores soportados por TexToES. Por ejemplo, $\text{suma} \in \mathbb{OP}$

Entonces, VAR en la gramática puede ser involucrar una producción del tipo:

$VAR = \mathbb{R}, \mathbb{T}, \mathbb{OP}$

La gramática es fácil de extender, por lo tanto, si se quiere añadir soporte para algún operador no soportado, con simplemente describirla en la gramática será suficiente.

Las plantillas o templates son una forma de centralizar la representación del texto origen (source) al texto objetivo (target). Y saca su mayor provecho cuando la representación es uno a uno. Si hubiéramos seguido un approach basado en PMathML en lugar de CMathML, donde para cada PMathML podíamos tener distintas interpretaciones una gramática template-based no nos hubiera ayudado demasiado. Ya que generaríamos, algunas veces, resultados incorrectos o mal interpretados.

La gramática completa puede consultarse en el anexo de éste documento.

Chapter 4

Construcción y evaluación de la gramática

4.1 Reglas descriptas en la gramática

Como se mencionó en la sección anterior, se puede utilizar un generador de lenguaje natural o verbalizador basado en plantillas (template-based NLG). Es necesario en éste punto construir templates que representen la manera más usual de leer expresiones matemáticas. Para ello se contactó a distintos estudiantes de carreras matemáticas, e incluso Ingenieros en Sistemas y Lic. en Cs de computación para poder reconocer las distintas maneras de leer una expresión. Todas las representaciones fueron tomadas en cuenta, dado que todas ellas describen las expresiones matemáticas dependiendo de quien la puede leer.

Las fórmulas matemáticas pueden leerse dependiendo del contexto geográfico, de la edad, de la convención utilizada en la institución educativa, de distintos libros se puede llegar a la conclusión que no hay una única manera de representar mediante texto una misma fórmula matemática. Por ejemplo,

$$\frac{a}{b}$$

Puede leerse de distintas formas, por ejemplo, a sobre b, a dividido b, la fracción de a y b, a partido por b. Por lo tanto, es necesario conocer las distintas alternativas de

la gramática.

Dado que se utiliza un template como gramática de generador de lenguaje, es fácil de añadir y extender a nuevas producciones si alguna no se encontrase disponible o las disponibles son incorrectas.

4.2 ¿El resultado es la representación más usual?: Evaluación

La evaluación de estos sistemas es un campo muy vital tanto para determinar cuán efectivo es nuestro sistema y además de optimizarlo en caso de alguna oportunidad de mejora.

Muchas de las propiedades que conforman a la calidad de una traducción son:

- **Fidelidad**, busca responder a la pregunta: ¿la transcripción tiene el mismo significado que el idioma fuente?
- **Comprensibilidad**, una transcripción puede ser correcta pero sin embargo difícil de entender, a veces más difícil que el idioma fuente.
- **Fluidez**, la traducción puede tener el significado correcto y hasta fácil de entender, sin embargo, podría suceder que ningún usuario la menciona alguna vez.

Para evaluar TexToES se han encarado tres tipos de evaluaciones, aunque las tres en conjunto buscan medir y evaluar el sistema para adaptarse a mejores resultados y optimizaciones.

Se han desarrollado y usado métodos de evaluación que son humanos, automáticos y semi-automáticos (humanos presentes o automatización asistida).

Antes de comenzar el desarrollo de cualquier tipo de evaluación y también debido a la inexistencia de reglas para leer expresiones matemáticas, se ha realizado una encuesta que su objetivo fue el de reconocer las representaciones más usuales para distintas fórmulas matemáticas. Las encuestas consisten en mostrar al encuestado 3

formulas matemáticas y el sujeto debía de responder a la siguiente consigna:

Si tuvieras que dictar las siguientes formulas a alguien para que las escriba, ¿Cómo las dictarías? La consigna es que escribas una descripción en texto de cómo leerías las siguientes formulas.

Por ejemplo, si la formula planteada es " $(a + 2)/3 = f$ ", podrías escribir algo así como " $((a \text{ más } 2) \text{ dividido } 3) \text{ es igual a } f$ ".

- Escribir los numeros y formulas tal cual se presentan, como en el ejemplo. - Agrupar con parentesis el orden en que lo dictarías. - Escribir "(" en lugar de "abrir parentesis" - Si no sabes como escribir una formula, simplemente omitila.

Las encuestas se dividían por niveles, donde cada nivel conenía distintos tipos de expresiones. En el nivel número uno, podmiamos encontrar operaciones básicas como la suma, división, multiplicación, exponentes, raíces. En el nivel dos, podíamos encontrar ya expresiones trigonométricas, funciones, derivadas, integrales. En el nivel tres, podíamos encontrar sumatorias, productorias, matrices, etc.

De esta forma podemos aplicar un patrón de limpieza, donde no diferenciamos números, ni funciones, ni variables para poder quedarnos con producciones generales, y conocer desde distintas entradas recibidas en las encuestas cuáles son las posibles verbalizaciones para las distintas expresiones matemáticas.

Las encuestas han sido publicadas en distintos grupos y listas de distribución de FaMAF, y también ha sido difundida entre amigos y compañeros, alumnos y demás.

4.3 Descripción del procedimiento

El procedimiento consistió de distintas etapas, todas necesarias para evaluar la transcripción obtenida.

- Recolección de datos: Construcción y difusión del medio utilizado para obtener datos.
- Pre-procesamiento de datos: Llevar los datos a un nivel general, para no distinguir entre números o nombres de variables. Además de estandarizar las respuestas para su fácil procesamiento.
- Armado de Corpus (Corpus building): Reunión de los datos obtenidos, añadiendo información relevante tal como etiquetado y cantidad de ocurrencias.
- Constructor de evaluadores: Construcción de distintos evaluadores que indiquen una métrica de confianza de la transcripción final. Tenemos un evaluador automático, semi-automático y manual.
- Recolección de resultados: Presentación de resultados obtenidos y acciones tomadas.

4.3.1 Recolección de datos

Se ha utilizado **Google Forms** para crear los formularios que recolectaron la información necesaria que evidencie la forma de leer las fórmulas matemáticas.

Ha sido un total de 45 preguntas que abarcaron distintos niveles de la matemática, (CITA A: my Goole forms) desde los operadores básicos como la suma y resta, hasta diferenciales. Un total de tres niveles y cinco encuestas cada uno presentando tres fórmulas.

Se ha recibido un total de 1298 respuestas de los distintos niveles. La tercera parte de las respuestas caen dentro del primer nivel. Mientras que último nivel ha obtenido menos de 100 respuestas en todas sus encuestas.

Todas las respuestas obtenidas pueden ser consultadas en el repositorio de ésta tesis organizados por su respectivo nivel en un archivo de formato csv (comma separated values).

Cada archivo csv cumple con la siguiente estructura:

Una respuesta de un usuario está separada por comas (,) y el delimitador son las comillas dobles ("). Esto significa que todo caracter de **coma** dentro de dos comillas dobles no denotará una respuesta distinta.

En la primer linea estan presentes los títulos de las preguntas y un dato adicional agregado por Google Forms que es el timestamp.

Ejemplo:

"Timestamp", "¿Cómo la escribirías? n1.e2.f1", "¿Cómo la escribirías? n1.e2.f2", "¿Cómo la escribirías? n1.e2.f3"

En las demás líneas se encuentran las respuestas.

Ejemplo:

"2015/11/24 9:07:45 PM GMT-3", "r1", "r2", "r3"

De ésta forma podemos encontrar en los archivos csv en su primer columna la fecha en la que el usuario ha respondido, en su segunda columna la respuesta asociada a la primer pregunta, y así sucesivamente.

Una respuesta vacía implica la presencia de dos comillas dobles juntas ("").

4.3.2 Pre-pocesamiento de datos

Con el pre-procesamiento se intenta corregir o eliminar datos erróneos del conjunto de datos obtenidos. Tambien se buscó la presencia de datos incompletos, inexactos, etc. y luego substituir, modificar o eliminar estos datos sucios. Después de este pre-procesamiento, los datos podrán ser compatibles con el resto del sistema que evaluará la transcripción.

El pre-procesamiento se llevo a cabo teniendo en cuenta las siguientes consideraciones:

- Eliminar todo caracter acentuado. Por ejemplo, el wording asociado al operador suma que puede ser **más** o **mas** a fines prácticos es el mismo.

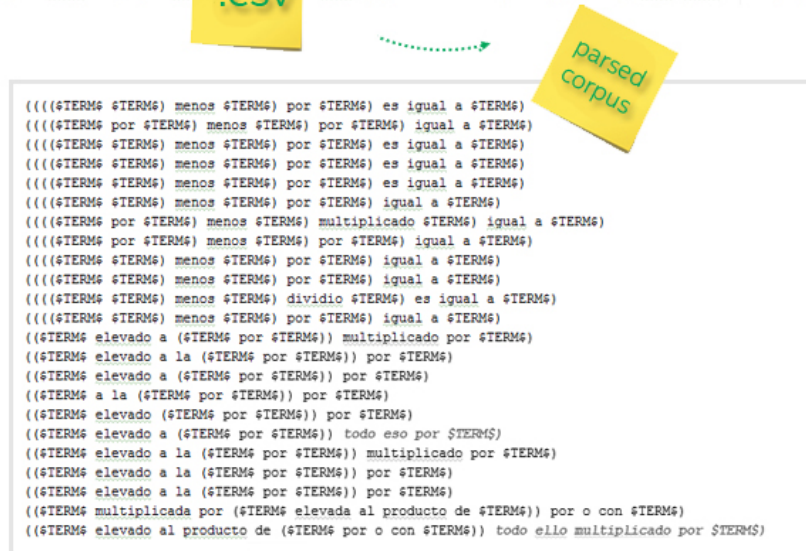
- Con la misma razón anterior, se reemplazaron los nombres de variables, constantes y números por la palabra **\$VAR\$**. Porque para la verbalización de la suma, por ejemplo, podemos tener las siguientes instancias: 6 mas 5, 4 mas 9 y 7 sumado a 6. Nos interesa contar ambas primeras como la misma verbalización y diferenciarla de la segunda por contener otro wording.
- Corregir paréntesis para los casos que éstos no esten balanceados, no esten presentes o esten mal ubicados.
- Identificación de multiples respuestas en una misma respuesta de formulario. Para estos casos, solo se trata cada respuesta como una instancia distinta.

Este comportamiento está implementado dentro de la clase *language_evaluator* en el método *prepare_corpus* dentro del código fuente de éste proyecto.

El método *prepare_corpus* se encarga de revisar los archivos .csv obtenidos desde Google Forms, ejecutar las consideraciones descriptas anteriormente y crear un archivo llamado **parsed_responses.txt** con los datos limpios.

Para ver gráficamente su funcionamiento observemos la siguiente figura:

```
(tres equis menos 2) por siete igual a cero", "equis a la $TERM$ por b por c.", "raiz sexta de cinco",
tres equis menos 2 por siete, igual a cero", "equis, exponente $TERM$ por b, por ce", "raiz seis de ci
((3 equis menos 2) multiplicado por 7) es igual a 0", "(equis elevado a: $TERM$ por b) multiplicado
((3 x menos 2) por 7) es igual a 0", "(x elevado a ($TERM$ por b)) multiplicado por c", "(raiz sexta d
(3 X menos 2) multiplicado por 7 es igual a 0", "(X elevado a $TERM$ por b) multiplicado por c", "Raiz
(3 X menos 2) multiplicado por 7 es igual a 0", "(X elevado a $TERM$ por b) multiplicado por c", "Raiz
(((3 por x) menos 2) por 7) igual a 0)", "(x elevado a la $TERM$ por b) por c)", "raiz sexta de cinc
el termino tres equis menos dos multiplicado por siete es igual a cero", "el termino equis elevado a
el termino 3 por equis menos 2 finalmente multiplicado por 7 es igual a cero", "el termino equis elev
(3 por x menos 2) por 7", "x elevado a la $TERM$ por b, por c", "raiz sexta de 5", "", ""
(Tres de X menos 2) por si", "X elevado a la $TERM$ por b esa potencie multiplicada por C "
((3 x menos dos) por si", "a cero", "(X a la $TERM$ por b) por ce", "Raiz sexta de 5", "", ""
```



```
((($TERM$ $TERM$) menos $TERM$) por $TERM$) es igual a $TERM$)
(((($TERM$ por $TERM$) menos $TERM$) por $TERM$) igual a $TERM$)
(((($TERM$ $TERM$) menos $TERM$) por $TERM$) es igual a $TERM$)
(((($TERM$ $TERM$) menos $TERM$) por $TERM$) es igual a $TERM$)
(((($TERM$ $TERM$) menos $TERM$) por $TERM$) es igual a $TERM$)
(((($TERM$ $TERM$) menos $TERM$) por $TERM$) igual a $TERM$)
(((($TERM$ por $TERM$) menos $TERM$) multiplicado $TERM$) igual a $TERM$)
(((($TERM$ por $TERM$) menos $TERM$) por $TERM$) igual a $TERM$)
(((($TERM$ $TERM$) menos $TERM$) por $TERM$) igual a $TERM$)
(((($TERM$ $TERM$) menos $TERM$) por $TERM$) igual a $TERM$)
(((($TERM$ $TERM$) menos $TERM$) dividido $TERM$) es igual a $TERM$)
(((($TERM$ $TERM$) menos $TERM$) por $TERM$) igual a $TERM$)
(($TERM$ elevado a ($TERM$ por $TERM$)) multiplicado por $TERM$)
(($TERM$ elevado a la ($TERM$ por $TERM$)) por $TERM$)
(($TERM$ elevado a ($TERM$ por $TERM$)) por $TERM$)
(($TERM$ a la ($TERM$ por $TERM$)) por $TERM$)
(($TERM$ elevado ($TERM$ por $TERM$)) por $TERM$)
(($TERM$ elevado a ($TERM$ por $TERM$)) todo eso por $TERM$)
(($TERM$ elevado a la ($TERM$ por $TERM$)) multiplicado por $TERM$)
(($TERM$ elevado a la ($TERM$ por $TERM$)) por $TERM$)
(($TERM$ elevado a la ($TERM$ por $TERM$)) por $TERM$)
(($TERM$ multiplicada por ($TERM$ elevada al producto de $TERM$)) por o con $TERM$)
(($TERM$ elevado al producto de ($TERM$ por o con $TERM$)) todo ello multiplicado por $TERM$)
```

FIGURE 4.1: Funcionamiento de prepare_corpus

4.3.3 Corpus building

Esta parte del procedimiento se encarga de construir lo que será el corpus que se utilizará para hacer las evaluaciones de las transcripciones. En el corpus se reúne toda la información que los contribuyentes dieron en sus respuestas. Es información limpia y que puede exportarse para utilizarla en otro proyecto.

El comportamiento que describiré en esta sección puede encontrarse en el método `parse_corpus` dentro de la misma clase.

Para crear el corpus simplemente se ha implementado un parser que sabe interpretar los datos limpios obtenidos por el punto anterior, para recolectar en un solo lado toda la información.

Este método se encarga de extraer de cada respuesta todas las estructuras gramaticales compuestas en él. Explicaremos esto mediante el siguiente ejemplo, supongamos:

input: (($\text{\$TERM\$}$ elevado a ($\text{\$TERM\$}$ por $\text{\$TERM\$}$)) multiplicado por $\text{\$TERM\$}$)
 output: []

Nos posicionaremos entre el último parentesis "(" y el primer paréntesis ")" a partir de él. Y extraeremos eso como una regla gramatical atómica, reemplazandolo en la fórmula original como $\text{\$TERM\$}$.

input: (($\text{\$TERM\$}$ elevado a $\text{\$TERM\$}$) multiplicado por $\text{\$TERM\$}$)
 output: [($\text{\$TERM\$}$ por $\text{\$TERM\$}$),]

Repetimos el algoritmo anterior obteniendo:

input: ($\text{\$TERM\$}$ multiplicado por $\text{\$TERM\$}$)
 output: [($\text{\$TERM\$}$ por $\text{\$TERM\$}$), ($\text{\$TERM\$}$ elevado a $\text{\$TERM\$}$)]

El algoritmo finaliza cuando lo que encierran los parentesis seleccionados es el total de la cadena. Por lo tanto el resultado que obtenemos es:

output: [($\text{\$TERM\$}$ por $\text{\$TERM\$}$), ($\text{\$TERM\$}$ elevado a $\text{\$TERM\$}$), ($\text{\$TERM\$}$ multiplicado por $\text{\$TERM\$}$)]

De ésta manera pudimos extraer de una sola respuesta, gracias a tener los datos limpios, tres distintas reglas gramaticales que aportan a la construcción del evaluador final.

Finalmente, este proceso es aplicado a cada respuesta recibida obteniendo un abundante listado de reglas gramaticales que formaran parte de las evaluaciones.

¿Cómo está organizado el corpus de reglas de gramaticales?

Este corpus que reglas gramaticales, está compuesto por todas las reglas atómicas que se extrajeron en el procesamiento anterior. Cada regla gramatical denota una forma de escribir una sola operación matemática. Por ejemplo, si el operador en cuestión fuese la suma (que en MathML es tratado como plus) podemos obtener como reglas gramaticales atómicas los siguientes ejemplos (**$\text{\$VAR\$}$ sumado a $\text{\$VAR\$}$**) o (**$\text{\$VAR\$}$ mas $\text{\$VAR\$}$**) o aún así (**la suma de $\text{\$VAR\$}$ y $\text{\$VAR\$}$**).

Luego, como paso final en todo éste procesamiento. Para fines de performance, se busca persistir en un solo archivo información pre-calculada y necesaria para computar las evaluaciones.

Cada regla gramatical atómica está asociada a una etiqueta MathML que denote la relación directa entre la etiqueta y su verbalización. Dado que podemos tener distintas ocurrencias de la misma regla gramatical y la etiqueta, se decidió añadir a esta version sumariada del corpus un número que denote la cantidad de ocurrencias de una misma regla.

Por lo tanto, en conclusión, el siguiente es un extracto de éste corpus mencionado:

```
root,42,(raiz $TERM$ de $TERM$)
function,70,($TERM$ de $TERM$)
times,17,($TERM$ multiplicado por $TERM$)
times,168,($TERM$ por $TERM$)
exp,45,($TERM$ al $TERM$)
exp,59,($TERM$ a la $TERM$)
```

Mediante él, podemos obtener respuestas a distintas preguntas que nos podamos hacer. Por ejemplo, ¿De cuántas maneras distintas regularmente se puede referir a la potencia?, ¿Qué regla es para la multiplicación es mas frecuente?, ¿Hay alguna operación que se diga de la misma forma?.

4.3.4 Constructor de evaluadores

En el contexto de este artículo, una métrica es un indicador que evalúa la producción final o transcripción y representa la calidad de la salida.

La calidad de una traducción es inherentemente subjetiva, no hay ningún objetivo cuantificable o bueno. Por lo tanto, cualquier métrica debe asignar puntuaciones de calidad relacionadas con las que daría un humano. Es decir, una métrica debería puntuar alto a transcripciones que los humanos puntuen altamente, y dar puntuaciones bajas a aquellas que los humanos den puntuaciones bajas. El juicio humano es el punto de referencia para evaluar las métricas automáticas.

Este sistema define tres tipos de métricas que indicaran el grado de confianza que tendrá el usuario.

Evaluador automático

Un evaluador automático es aquel que no necesita de la presencia de un humano para poder evaluar, es una métrica que se desarrolla y se ejecuta desatendidamente. Ésta métrica se enfoca en medir **la fluidez de las transcripciones**.

Dado que para medir la fluidez no necesitamos conocer el lenguaje de origen (en el entorno de ésta tesis serían las fórmulas matemáticas) y si conocer el lenguaje destino (transcripciones) esta métrica es desarrollada y ejecutada gracias a la presencia de los datos recolectados de las encuestas.

Por cada transcripción que el usuario consulte, se le informará el grado de confianza de esa transcripción. Para ayudarnos a entender qué significa el grado de confianza, enunciaré algunas definiciones.

Definición:

El *conjunto de las transcripciones atómicas*, es el conjunto de todas las reglas gramaticales atómicas que llevaron a producir la transcripción. Por ejemplo:

Para el caso de x^{ab} tenemos que su transcripción asociada es: (X elevado a la (A multiplicado por B)) entonces el conjunto de las transcripciones atómicas es el conjunto que contiene a \$TERM\$ multiplicado por \$TERM\$ y (\$TERM\$ elevado a la \$TERM\$). Dado que solo ambas reglas gramaticales han sido usadas para producir la transcripción.

Definición:

La *etiqueta asociada a una transcripción*, denotada como $tag(t_i)$, es la etiqueta MathML para la cual tiene asociada tal transcripción atómica.

Para el caso de (\$TERM\$ mas \$TERM\$) el tag asociado es *plus*.

Con estas definiciones estamos listos para definir la métrica que queremos mencionar.

Definición:

Sea trx una transcripción obtenida por el verbalizador, se define como **grado de confianza** a:

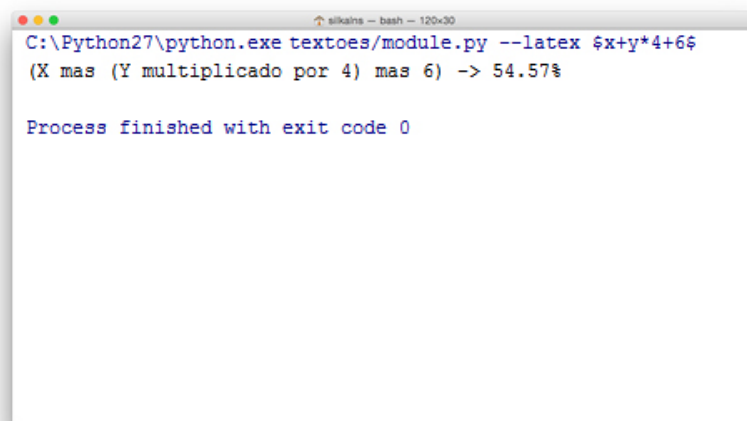
$$GDC(trx) = \prod_{t_i \in trx} \frac{count(t_i)}{count(tag(t_i))}$$

Dónde se denota t_i como las transcripciones atómicas dentro de trx .
 $count(t_i)$ es la cantidad de ocurrencias de la t_i en el corpus sumariado.
 $count(tag(t_i))$ es la cantidad de transcripciones que tiene el tag asociado a la t_i .

Esta métrica es un número entre 0 y 1, que intenta mostrarle al usuario un número que represente que tan frecuente es la transcripción con respecto a los datos obtenidos por los usuarios mediante los formularios.

Dado que todos los datos que el cálculo de GDC pide están presentes en el corpus de información externa, hace que esta métrica pueda ser calculada automáticamente.

Actualmente TexToES está mostrando por cada transcripción su versión percentil de la métrica ($GDC(trx) * 100$)



```

C:\Python27\python.exe textoes/module.py --latex $x+y*4+6$
(X mas (Y multiplicado por 4) mas 6) -> 54.57%

Process finished with exit code 0

```

FIGURE 4.2: GDC de una transcripción

Evaluador semi-automático (human-in-the-loop)

Este tipo de evaluadores semi-automáticos (HITL o human-in-the-loop) son evaluadores que requieren la interacción del humano para poder proceder.

La idea detrás de éste método es ver como se comporta TexToES ante una posible traducción inversa. El procedimiento fue el siguiente:

- Se han seleccionado 20 fórmulas latex y se ha estimulado TexToES con ellas obteniendo 20 transcripciones.
- Se ha construido una encuesta con estas 20 transcripciones y se les ha pedido a participantes contribuyentes que escriban la fórmula matemática asociada a esa transcripción.
- Por cada respuesta, se ha reconstruido la formula latex.
- Se compara de igual a igual las fórmulas latex obtenidas con las fórmulas latex empleadas para obtener las transcripciones

Los datos empleados (latex y transcripción) en la encuesta pueden ser encontrados en el Apéndice de ésta tesis, como así también las respuestas de los contribuyentes.

Ésta es la parte donde el humano participa en la evaluación, el resto de la evaluación es automática pero necesita de recolectar resultados a través de humanos.

Por lo tanto queremos ver cómo se comporta TexToES informando una transcripción, vamos a definir para este tipo de evaluación la siguiente métrica GDC.

$$GDC = \sum_{i=1}^N \frac{1}{N^2} * count(latex_i == resp_i)$$

Dónde N es la cantidad de transcripciones evaluadas, en éste caso fueron 13.

$latex_i$ es el i-ésimo latex empleado para obtener la transcripción.

$resp_i$ es la respuesta i-ésima (que también es un latex) por parte de los contribuyentes.

Cabe mencionar que esta metrika de confianza no es una métrica que habla de una transcripción, sino mas bien es una métrica que habla de la confianza del sistema

de traducción. La métrica intenta capturar la idea de la proporción de un promedio de respuestas correctas.

Evaluator humano

Se han presentados dos evaluadores que buscan definir la calidad de la transcripción pero, sin embargo, la calidad de la traducción es algo subjetivo y de cierta manera el Gold Standard es tener un juicio humano sobre nuestro sistema. Dado que la salida de éste sistema es para un ser humano, es correcto pensar que un ser humano lo evalúe. Por otro lado, en la instancia de evaluación humana están presentes factores propios de la comprensión que le permite al juez identificar posibles errores en la traducción.

Pero para ello se necesita de tener un juez humano que entienda tanto del tipo de entrada (**source**) como el tipo de salida (**target**) para poder hacer el juicio subjetivo sobre la traducción, esto implica hacer un juicio subjetivo indicando la calidad de una transcripción para una fórmula matemática dada.

La técnica empleada de medición aquí fue presentar una serie de fórmulas matemáticas con sus respectivas transcripciones y se le pide al juez calificar utilizando una escala de múltiples puntos. Estas calificaciones requieren de un juez que entienda ambos lenguajes.

El juicio puesto en juego es de si **la transcripción es adecuada para la fórmula**.

Se ha utilizado Google Forms para hacer estas preguntas a los contribuyentes siguiendo la distinta presentación.

$$\sum_{x=a}^b f(x)$$

(la sumatoria de X que va desde A hasta B de (F de X)) *

☐ 3 - Completamente adecuado

☐ 2

☐ 1 - Tendencia a adecuado

☐ 0

☐ -1 - Tendencia a inadecuado

☐ -2

☐ -3 - Completamente inadecuado

FIGURE 4.3: Evaluador humano

Se han escrito un total de 20 preguntas que muestran las fórmulas matemáticas en imágenes junto a las transcripciones obtenidas con TexToES. Las 20 fórmulas matemáticas intentan contener todo el alcance de TexToES desde sumas, restas hasta sumatorias y aplicación de funciones. Incluso combibandolas.

Los resultados obtenidos se muestran en la siguiente sección.

4.3.5 Recolección de resultados

Hemos mostrado varias formas de medir TexToES que buscan estar alineados con el juicio humano, es decir, si un humano dice que la transcripción es buena queremos que nuestros evaluadores digan lo mismo, y así también con los casos en que las transcripciones sean malas. En esta sección se intenta enumerar algunos resultados obtenidos para intentar aplicar correcciones a TexToES para volverlo más real.

De los tres tipos de métricas que se han elaborado pudimos extraer sugerencias y conocer cómo es que el usuario final suele leer las distintas fórmulas matemáticas.

Para el caso de las encuestas que sirvieron para evaluar las transcripciones automáticamente, aquellas que se le pedía al contribuyente cómo leerían un conjunto de fórmulas matemáticas, se han extraído los siguientes patrones.

Patrones de traducción

- El 100% de los encuestados se refiere a la suma entre x e y como **X más Y**. Por lo tanto la regla de traducción para la etiqueta <plus> debería ser completada con la regla mencionada.
- Tanto para el caso de la potencia como para la raíz, el 100% de los encuestados utilizan **cuadrado** y **cubo** para referirse a la aplicación de esas funciones para el 2 y el 3 respectivamente.
- Cuando no es la raíz cuadrada ni cúbica, el 100% de los encuestados utiliza **raíz X de Y**.
- Cuando no es la exponente cuadrada ni cúbica, el 41% de los encuestados se refiere a **X a la Y** para mencionar la exponencia de X con Y . El 33% lo menciona como **X al Y** y el 26% lo menciona como **X elevado a Y**.
- Así como a la suma, para el caso de la resta el 100% de los encuestados dicen **X menos Y**.
- El 55% de los encuestados igualan x con y mencionando **X es igual a Y**, mientras que el resto dicen **X igual a Y**.
- Para el caso de la multiplicación tenemos que el 90% se refiere a la multiplicación entre x con y como **X por Y**, mientras que el 10% restante lo dice como **X multiplicado por Y**. Es importante notar aquí que TexToES prefiere traducir la multiplicación como *X multiplicado por Y* haciendo que elija la transcripción menos probable. Una acción a tomar a partir de aquí es cambiar la regla de traducción de TexToES para que elija la más probable.
- El 73% de los encuestados se refieren a la división entre x e y como **X sobre Y**, mientras que el porcentaje restante elige decir **X dividido Y**. Cambiar la regla de traducción de TexToES para este caso también hará cambiar las métricas.

Estos son algunos datos que podemos extraer de las encuestas realizadas para evaluar a TexToES automáticamente.

Algunas sugerencias por partes de los humanos entrevistados para el evaluador semi-automático fueron:

- Utilizar **A intersección B**, en lugar de **A intersectado por el conjunto B** para el caso de la intersección entre dos conjuntos.
- En la aplicación de la función F en el argumento X, mencionar **F de X** en lugar de **F evaluado en X**.

Es importante mencionar que estas sugerencias fueron tomadas para modificar la forma de traducir de TexToES, dado que coinciden con la extracción de patrones detalladas en el punto anterior.

Por último, por el lado de la evaluación con juicio humano presentaré aquí algunos resultados.

En total se han recibido una cantidad de **29 respuestas** para las 20 preguntas que presentaba el Formulario.

Recordemos que cada contribuyente podía evaluar la transcripción puntuando de la siguiente manera

- **3** *completamente adecuado*
- **2**
- **1** *tendencia a adecuado*
- **0**
- **-1** *tendencia a inadecuado*
- **-2**
- **-3** *completamente inadecuado*

Para ayudarnos a interpretar algunos resultados vamos a dividir la tabla de puntos anteriores en tres categorías. La categoría que contiene los puntajes 3, 2 y 1 serán llamados puntajes **buenos**, para los puntajes marcados como 0 será tratado como puntaje **neutro** finalmente, para los puntajes de -1, -2 y -3 serán llamados puntajes **malos**. Algunos datos interesantes que pueden observarse son:

- 9 de 20 transcripciones obtuvieron un puntaje de 3 (completamente adecuado) con más del 70%
- Solo 2 de 20 transcripciones obtuvieron un puntaje entre -3 y -2, y lo obtuvieron como 5%
- 9 de 20 transcripciones obtuvieron solo puntajes buenos
- 20 de 20 transcripciones obtuvieron en su mayoría de respuestas puntajes buenos

Podemos observar, con la siguiente imagen, que solo una transcripción de las 20 ha obtenido 3 como puntaje en todas sus respuestas asociadas.

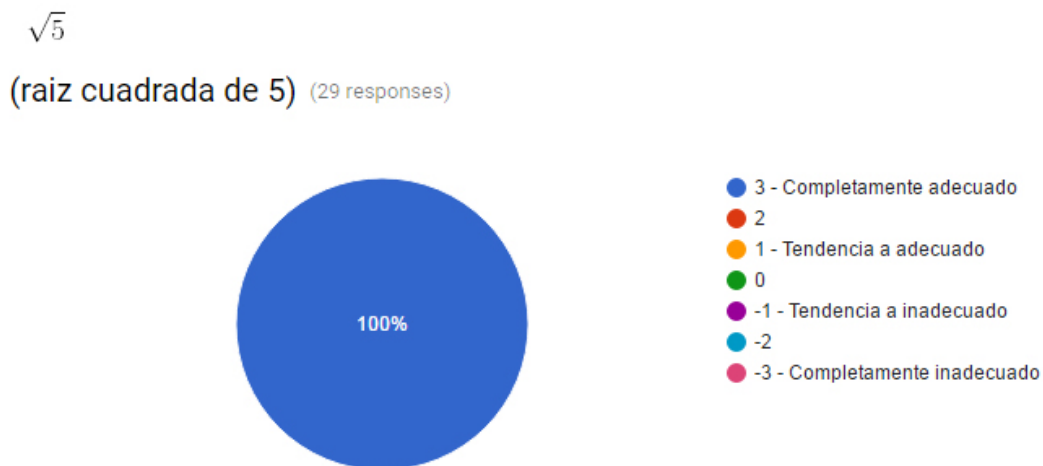


FIGURE 4.4: Transcripción con 100% de respuestas en completamente adecuado

El siguiente es un ejemplo de las 9 transcripciones que ha recibido solo puntaje bueno. Como se puede observar el 100 % de sus respuestas están en el conjunto de calificaciones buenas (1, 2 o 3)



FIGURE 4.5: Transcripcion con solo puntaje bueno

Y por último, el siguiente es el caso que mas respuestas variadas ha tenido y aún así en su mayoría ha obtenido un puntaje bueno. El 69 % de sus respuestas han incluido 1, 2 y 3 como calificación. Además este también es 1 de las 2 transcripciones que ha obtenido un 3 en el 5% del total de sus calificaciones.



FIGURE 4.6: Transcripcion con solo puntaje bueno

Usted puede consultar la lista completa de respuestas en el anexo de éste documento.

Appendix A

Appendix Title Here

Write your Appendix content here.

Referencias

- [1] Censo Nacional de Población, Hogares y Viviendas 2010 Instituto Nacional de Estadística y Censos, Argentina *Disponible* : *http* :
//www.indec.gov.ar/ftp/cuadros/sociedad/PDLP1014.pdf
- [2] Mathematical Markup Language (MathML) Version 3.0 2nd Edition
https://www.w3.org/TR/MathML3/ 1998-2014 W3C (MIT, ERCIM, Keio, Beihang)
- [3] SnuggleTeX (1.2.2), David McKain, University of Edinburgh
http://www2.ph.ed.ac.uk/snuggletex/documentation/generating-content-mathml.html
- [4] Disponible: *http://www2.ph.ed.ac.uk/snuggletex/UpConversionDemo*
- [5] Recurso online donde muestra la relacion entre MathML y TEX *Disponible* : *http* :
//tex.stackexchange.com/questions/57717/relationship-between-mathml-and-tex
- [6] The Wikipedia Math Accessor Leo Ferres University of Concepcion, Chile, 2006,
Disponible: *http://www.inf.udec.cl/josefuentes/files/pdf/MathAcc-ET.pdf*
- [7] Pipelines, Templates and Transformations: XML for Natural Language Generation Graham Wilcock University of Helsinki,, Finland Disponible:
http://www.ling.helsinki.fi/gwilcock/Pubs/2001/NLPXML-01.pdf
- [8] Real vs. template-based natural language generation: a false opposition? Kees van Deemter y Emiel Krahmer, University of Brighton Tilburg Disponible:
http://doc.utwente.nl/65551/1/templates-squib.pdf
- [9] GALE: Part 5, Machine Translation Evaluation Bonnie Dorr, Matt Snover, Nitin Madnani
Disponible: *https://www.cs.cmu.edu/alavie/papers/GALE-book-Ch5.pdf*