UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

INDEXACIÓN Y BÚSQUEDA DE EXPRESIONES MATEMÁTICAS A GRAN ESCALA EN CORPUS MASIVOS DE DOCUMENTOS IMPRESOS

MÁSTER UNIVERSITARIO EN INTELIGENCIA ARTIFICIAL,
RECONOCIMIENTO DE FORMAS E IMAGEN DIGITAL

Autor
Ernesto Noya Garcia

Directores
José Miguel Benedi
Joan Andreu Sánchez

Curso 2019/2020

Resumen

En la actualidad existen grandes cantidades de documentos científicos impresos, muchos de los cuales incluyen expresiones matemáticas. La búsqueda de información textual en estos documentos es una posibilidad ampliamente explotada por los motores de búsqueda más utilizados. Sin embargo, la búsqueda mediante consultas en forma de expresiones matemáticas es un campo apenas explorado. Los planteamientos actualmente más utilizados para abordar este problema se basan fundamentalmente en la búsqueda por similitud entre las imágenes, lo cual es inviable para búsqueda en colecciones masivas dado el elevado coste computacional de dichas aproximaciones.

En este proyecto se propone desarrollar un marco estadístico que facilite la indexación y búsqueda de expresiones matemáticas en grandes colecciones de imágenes digitalizadas. Debido a los problemas de segmentación y a la propia ambigüedad que puede existir en las expresiones matemáticas, los modelos que permitirán construir los índices de la colección y que permitirán representar la consulta se basarán en métodos estocásticos estructurales capaces de dar cuenta de la ambigüedad que puede surgir en el proceso de reconocimiento. Entre los retos que presenta este proyecto se encuentra la preparación de índices que incluyan medidas de confianza, estructuras de datos en forma de árbol sintáctico para realizar búsquedas estructurales y el aprendizaje automático discriminativo de modelos estructurales.

Índice general

Capítulo 1

Introducción

1.1. Motivación

Hoy en día, buscar información científica en textos electrónicos usando navegadores web es una actividad usual en el quehacer diario de los investigadores, escolares, estudiantes, archivistas y el público general, y análogamente localizar información científica en artículos y libros que están ubicados en servidores privados pertenecientes a universidades, editoriales, librerías, etc. es una actividad usual en muchas disciplinas científicas. La existencia de potentes motores de búsqueda integrados en los buscadores de mayor éxito actuales (google, yahoo, bing, ...) ha propiciado y facilitado al usuario la satisfacción de esta necesidad en la mayor parte de los casos. Estos motores de búsqueda se basan fundamentalmente en un preproceso de enormes colecciones de documentos que permite crear índices para facilitar posteriormente una búsqueda rápida y eficiente. Dichos índices se generan fundamentalmente a partir de datos textuales asociados a los documentos que son objeto de búsqueda por parte del usuario: páginas web, imágenes acompañadas de texto, etc.

En las últimas décadas hemos visto una digitalización masiva de do-

cumentos que residían en repositorios institucionales públicos y privados (archivos, hemerotecas y bibliotecas) de todo el mundo. Estos documentos digitales son puramente imágenes sobre las cuales no es posible realizar búsquedas en su contenido. Cuando las colecciones en las que se pretende realizar la búsqueda no vienen acompañadas con datos textuales entonces es necesario procesar convenientemente los datos para facilitar la creación de índices de búsqueda. Por ejemplo, cuando se desea buscar en una colección de imágenes que no tiene información textual asociada, usualmente se genera una información textual de manera automática obteniendo alguna descripción de la imagen y posteriormente se generan los índices que permiten realizar la búsqueda de manera eficiente.

Entre los varios esfuerzos para digitalizar el contenido de los distintos tipos de nuevas colecciones encontradas, queremos citar el enfoque del proyecto HIMANIS, que ha investigado con éxito cómo hacer que se pueda buscar eficientemente en una colección de documentos manuscritos antiguos compuesta por más de 70.000 imágenes para los cuales no existe transcripción de texto. [29].

De las colecciones de documentos en las que actualmente sigue sin ser posible realizar búsquedas, en este trabajo nos centraremos en expresiones matemáticas (EM) dentro de colecciones de documentos científicos impresos. Actualmente existen enormes cantidades de documentos impresos tales como artículos científicos de diferentes campos de conocimiento depositados en infinidad de repositorios públicos (p.e. arXiv) que incluyen usualmente númerosas expresiones matemáticas, escritas con editores de texto especializados (p.e. LaTeX). Cada uno de los cuales tiene su propio lenguaje para codificar EM y herramientas para renderizarlas en imágenes con varios formatos posibles.

Una alternativa utilizada para resolver este problema es desarrollar técnicas que busquen EM de acuerdo a la imagen renderizada de la EM en vez de la versión codificada. Sin embargo métodos basados en comparar imágenes no son realistas para la búsqueda en colecciones masivas dado

el alto coste computacional que conllevan. Si el problema de comparación se plantea de esta manera, entonces la búsqueda de EM en un documento electrónico (pdf) se asemeja al problema de buscar un objeto dado en una imagen (p.e. un automóvil en una calle). Pero una diferencia importante con el escenario previo es que en el caso de EM, dos EM que son escritas con símbolos diferentes o en distinto orden pueden tener el mismo significado y consecuentemente ser la misma EM. Por ejemplo si vemos las siguientes expresiones:

$$\sum_{i=1}^{\infty}(\tfrac{1}{i+j})^2 \quad \sum_{j=1}^{\infty}(\tfrac{1}{j+i})^2 \quad \sum_{j=1}^{\infty}(1/(j+i))^2 \quad \sum_{j=1}^{\infty}1/(j+i)^2 \quad \sum_{i=1}^{\infty}(\tfrac{1}{i+j})^2$$

Figura 1.1: Distintas maneras de escribir la misma expresión matemática.

Observamos que todas representan el mismo concepto. En estos ejemplos, tanto *i* como *j* son variables intercambiables que no modifican el significado de la expresión. Notar además que comparar todas las imágenes con técnicas tradicionales de búsqueda de imágenes mostraría diferencias dado los cambios de ubicaciones. Estos problemas en la búsqueda de EM pueden aliviarse por el contexto alrededor de la EM usando un modelo sintáctico que sea capas de trabajar con ambigüedad. Ésta otra alternativa permite la búsqueda en colecciones de documentos científicos impresos con consultas en formato de EM, pero es una área de investigación menos explorada.

1.2. Reconocimiento de expresiones matemáticas

Las expresiones matemáticas pueden aparecer formando parte del propio texto o bien de manera aislada dentro del documento. Como hemos visto, un problema inherente a las expresiones matemáticas es que pueden contener ambigüedad en diferentes niveles. Así, una misma expresión se puede escribir de distintas maneras y seguir expresando la misma operación. Este problema en búsqueda de expresiones matemáticas aparece

también en otros ámbitos: búsqueda de expresiones matemáticas en páginas xml (wikipedia), en repositorios públicos que incluyen transparencias (SlideShare), o bien en curso online que incluyen expresiones matemáticas (OCW). Por tanto, resulta de indudable importancia desarrollar técnicas efectivas que puedan tolerar la ambigüedad y permitan buscar expresiones matemáticas de manera automática en grandes colecciones de documentos.

El reconocimiento de patrones de EM, usualmente se distingue entre expresiones online y offline. Formulas offline son representadas como imágenes y pueden ser impresas o manuscritas. Las expresiones online son codificadas como una secuencia de puntos en el espacio y por lo tanto incluyen información temporal. Las expresiones manuscritas presentan más variación que las impresas, mientras muestras generadas online pueden generar mejores resultados que las offline dado que contienen información adicional [24].

El reconocimiento de notación matemática es tradicionalmente divido en 3 problemas [3, 35]: segmentación de símbolos, reconocimiento de símbolos y análisis estructural. El la literatura actual, existen dos enfoques principales: soluciones secuenciales y soluciones integradas. Las soluciones secuenciales tienden a buscar primero la mejor segmentación de la expresión de entrada en símbolos matemáticos. El análisis de la estructura es realizado luego sobre la mejor segmentación de símbolos obtenida [36]. Este tipo de solución no es capaz de resolver errores realizados durante la primer parte si las decisiones tomadas no son revisables. Las soluciones integradas tratan de usar información global de la expresión matemática para obtener la estructura final de la fórmula como un todo [2, 39]. La segmentación de símbolos y posterior reconocimiento son obtenidos como producto de la optimización global. Estos enfoques parecen más apropiados dado que obtener la mejor segmentación de símbolos depende de la estructura de la expresión, y viceversa

En este trabajo presentamos una solución integrada para el reconocimiento de expresiones matemáticas digitales offline. Este enfoque está ins-

pirado en el usado en offline handwritten text recognition (HTR) [19]. En HTR, el proceso de reconocimiento es modelado en distintos niveles de percepción integrados en un modelo único: modelos ópticos (Modelos ocultos de markov [28] o redes neuronales recurrentes [23]) son usados para modelar caracteres a nivel bajo; modelos de estado finito son usados para modelar palabras en el nivel medio [28]; y modelos n-gramas son usados para modelar el lenguaje a nivel alto.

Este trabajo sigue un enfoque análogo con distintos niveles de percepción integrados en un único modelo. Describimos un modelo estadístico basado en gramáticas bidimensionales probabilísticas libres de contexto (GIP-2D) en el nivel más alto dado que constituyen un modelo natural y potente para lidiar con problemas estructurales. Como nivel medio utilizamos símbolos matemáticos, construidos como un conjunto de componentes conectadas, siendo éstas las primitivas en el nivel más bajo.

Definimos el algoritmo de análisis sintáctico asociado que determina globalmente la expresión matemática basado en varias fuentes de información. Los aspectos que nos permiten obtener la segmentación y reconocimiento de símbolos como producto se explican en detalle.

En el enfoque integrado, el espacio de búsqueda se vuelve demasiado costoso y por lo tanto presentamos también técnicas basadas en información espacial y geométrica para reducir efectivamente este espacio, imponiendo restricciones basadas en la distancia entre trazos. Finalmente intentaremos hacer una estimación de todas las distribuciones probabilísticas.

1.2.1. Reconocimiento de texto manuscrito e índices probabilísticos

El enfoque del proyecto HIMANIS consiste en obtener índices probabilísticos para imágenes de documentos en una fase de preparación preliminar. Este enfoque es libre de segmentación y no es exactamente reconoci-

miento de texto manuscrito [30].

La herramienta fundamental en este enfoque es el grafo de palabras (o caracteres) calculado con modelos de estado-finito, en donde los pesos son cuidadosamente manejados como probabilidades. Cada camino por el grafo de palabras representa una posible transcripción y/o segmentación alternativa [21]. El enfoque ha sido probado en escenarios reales y repositorios públicos han empezado a incluir esta solución a sus servicios. [1]

Un grafo de palabras se define de manera simple como un conjunto de varios miles de transcripciones textuales de una línea de texto que se condensan en un grafo. Dichos grafos de palabras se obtienen automáticamente mediante el uso de técnicas conocidas de Reconocimiento de Texto Manuscrito (RTM) [29]. Los esfuerzos realizados en el proyecto HIMANIS abren la puerta a investigar y desarrollar técnicas de generación de índices de búsqueda en colecciones de documentos que hasta ahora eran inimaginables.

En este trabajo, el concepto análogo a grafos de palabras cuando se usan GIP-2D son los *hipergrafos* [6], un árbol de análisis sintáctico estaría representado por un camino especificado en el grafo. Cuando una EM es analizada con una GIP-2D cada árbol de análisis representaría una posible interpretación de la EM.

1.2.2. Trabajos relacionados

El problema de reconocimiento automático de expresiones matemáticas ha sido estudiado por décadas [1]. Muchos enfoques han sido propuestos [5, 36, 16] pero desafortunadamente la mayoría no pueden comparase debidamente por la falta de datasets públicos y métricas estandarizadas.

Distintos enfoques han sido presentados para reconocimiento de expre-

[1] `http://prhlt-kws.prhlt.upv.es/fcr/`

siones matemáticas. Zanibbi y Blastein [36] reconocieron una expresión como un árbol, y propusieron un sistema basado en una secuencia de transformaciones del árbol. Eto y Suzuki [7] desarrollaron un modelo para reconocimiento de fórmulas impresas que calcula el árbol mínimo de una red representando la expresión. Shi et al. [25, 15] presentó un sistema donde segmentación de símbolos y reconocimiento eran resueltos simultáneamente basado en grafos. Después generaban distintos candidatos de símbolos para la mejor segmentación, y la expresión reconocida era calculada en el análisis de estructura final [26].

Dado que es conocida la estructura de notación matemática, muchos enfoques son basados en gramáticas dado que constituyen una forma natural de modelar este problema. De hecho, las primeras propuestas en reconocimiento de expresiones matemáticas fueron basadas en gramáticas [1, 5]. Desde entonces, diferentes estudios fueron desarrollados usando distintos tipos de gramáticas. Chan y Yeung [4], por ejemplo, usaron gramáticas de clausura definida, el modelo de Lavirotte y Pottier [13] esta basado en grafos de gramáticas, Yamamoto et al. [34] presento un sistema usando gramáticas incontextuales probabilísticas (GIP), y MacLean y Labahn [16] desarrollaron un enfoque usando gramáticas relacionales y sets difusos. En este trabajo nos enfocamos en modelos basados en GIP-2D.

El objetivo de este trabajo es proponer una formulación que sea, más o menos, respetuosa con lo que ahora está implementado. Esta formulación está extraída de [39, 37] y adaptada al problema de reconocimiento de expresiones matemáticas en documentos impresos.

1.3. Objetivos de la propuesta

En RTM, la generación de los índices para poder realizar la búsqueda en tiempo real es un proceso que se realiza offline. Es decir, como etapa previa al proceso de indexación debemos reconocer las imágenes de líneas

manuscritas y después generar los índices con los términos que aparecen en el grafo de palabras. Posteriormente se diseña un motor de búsqueda que realiza la búsqueda de una consulta a partir únicamente de los índices. Esta aproximación que ha resultado tan exitosa en el proyecto HIMANIS es la aproximación que se pretende seguir en este proyecto. Sin embargo los problemas que se esperan encontrar son completamente nuevos y tienen varias complejidades agregadas que describimos a continuación.

En primer lugar el REM offline está todavía lejos de alcanzar los resultados que actualmente se obtiene en RTM. La imprecisión en REM no viene solo dada por fallos de reconocimiento en los símbolos y estructuras sintácticas, sino también en problemas de segmentación en la expresión matemática. Por ello, los índices que se deben generar deben contener medidas de confianza que expresen la confianza que el sistema de REM tiene en el árbol de análisis obtenido . Los sub-árboles a indexar (que representan subexpresiones de la expresión principal) deben tener en cuenta dicha medida de confianza en el proceso de generación del índice. Posteriormente, para que la búsqueda se realice en tiempo real es necesario definir estructuras de datos eficientes para almacenar los índices tal como la búsqueda jerárquica usada en el proyecto HIMANIS[29].

El proceso que realiza el motor de búsqueda que se utiliza en para textos manuscritos se basa fundamentalmente en una comparación exacta entre la consulta de búsqueda (una palabra) y los términos indexados extraídos del grafo de palabras (que también son palabras). Este proceso de búsqueda no es válido para el caso de expresiones matemáticas puesto que en las expresiones matemáticas subyace una estructura que hace que dos expresiones con símbolos diferentes puedan representar la misma expresión. Considérese por ejemplo el caso ilustrado en la Figura 1.1. Por ello la comparación no puede basarse en la simple comparación de cadenas tal como sucede en indexación de texto manuscritos. El proceso de comparación debe considerar sub-estructuras que además llevan incorporadas medidas de confianza.

En este trabajo presentamos un modelo formal basado en distintos es-

tudios. Principalmente el sistema para reconocimiento de expresiones esta basado en el análisis de GIP-2D presentado en [39]. Ese modelo resuelve la segmentación de símbolos calculando las componentes conectadas de los trazos entrada y combinándolos utilizando producciones de la gramática (p.e. un signo igual se representa como una línea horizontal arriba de otra). Sin embargo, esta estrategia requiere considerar clases adicionales para la composición de símbolos (como una i sin punto o letras conectadas en una función) y encontrar buenas relaciones espaciales para su combinación. Más aun, no puede resolver símbolos cuyos trazo se cruzan o símbolos cuya composición no se haya tenido en cuenta con producciones especificadas en la gramática. Entonces, la segmentación no es una variable oculta sino que depende en decisiones tomadas en la gramática y la composición de símbolos. Además, en [39] no fue abordada la estimación de la GIP-2D.

Además, este trabajo investiga: i) La generación de los n-mejores árboles de análisis desde una GIP-2D de forma similar a como se explica en [11], ii) Cómo obtener hipergrafos desde los n-mejores árboles de análisis, y iii) Los algoritmos necesarios para normalizar el hipergrafo. Todos estos pasos son necesarios antes de obtener índices probabilísticos para imágenes de documentos.

1.4. Estructura de la memoria

Con el objetivo de facilitar la comprensión de esta memoria, presentamos los distintos modelos estudiados de forma que un modelo cuyo resultado sea necesario en una sección del trabajo será introducido antes de esa sección, viendo distintas definiciones del marco estadístico que proponemos a medida que sea necesario y respetando la notación introducida. Con este objetivo dividimos el resto de esta memoria en 4 capítulos donde cada uno hace uso de los conceptos presentado en los anteriores:

- En el siguiente capítulo seguimos describiendo el problema de REM,

presentamos los primeros componentes del marco estadístico junto las estructuras necesarias para implementarlo y los algoritmos para generar el árbol de análisis más probable para una EM.

- El tercer capítulo extenderá las definiciones, estructuras y algoritmos introducidos en el segundo para poder generar los N-Mejores árboles sintácticos dada una EM e introduce nuevas definiciones y algoritmos para combinar los árboles en una estructura compacta de hipergrafo y normalizarlo.
- En el cuarto capítulo presentamos una implementación de los distintos algoritmos propuestos en este trabajo y evaluamos experimentalmente el marco estadístico propuesto.
- En base a los resultados obtenidos en la evaluación, en el último capítulo resumimos los conocimientos adquiridos en este trabajo y discutimos el posible uso del sistema propuesto en futuros proyectos.

Capítulo 2

Reconocimiento de expresiones matemáticas en documentos impresos

Una vez presentada la motivación del proyecto y habiendo repasado los distintos trabajos que han abordado este problema. En este capítulo presentaremos los distintos modelos que utilizaremos a lo largo del trabajo para hacer un análisis estructural de una expresión matemática de entrada y discutiremos las ventajas que aportan cada uno.

2.1. Marco estadístico

En reconocimiento de expresiones matemáticas impresas (REM), dada una imagen, o una región dentro de ésta, el primer paso es considerar una función de representación para mapear las imágenes a otra representación. En nuestro caso, la función de representación extrae el conjunto de componentes conectadas $\boldsymbol{z} = \{z_1, \ldots, z_{|\boldsymbol{z}|}\}$, de la imagen de entrada.

Formalmente, sea O una imagen, o una región de una imagen, formada por la caja de mínima inclusión $[(x, y) \times (x + I, y + J)]$, donde (x, y) son las coordenadas del vértice inferior izquierdo e $I \times J$ son las dimensiones, en número de píxeles, de dicha región:

Definición 1 *Dada imagen o una región de una imagen O, definimos una componente conexa dentro de O como un área de pixeles continuos con valor positivo. Para evitar que el ruido existente en la imagen genere falsos positivos, requeriremos un tamaño en pixeles mayor a un umbral para las componentes que deben considerarse en las hipótesis.*

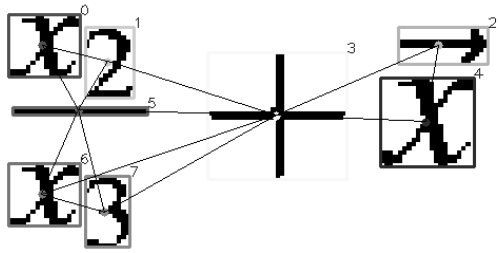


Figura 2.1: Conjunto de componentes conectadas y relaciones de visibilidad entre componentes para la imagen de entrada de la EM "$\frac{x_2}{x_3} + \vec{x}$".

Dado O, $\boldsymbol{z} = z_1, \ldots, z_N$ es la secuencia de *componentes conexas* extraídas de O. Donde $z_i[(x_i, y_i) \times (x_i + I_i, y_i + J_i)]$ es la caja de mínima inclusión de la i-ésima componente conexa de talla $I_i \times J_i$.

El problema de encontrar la estructura t asociada con O, se puede plantear entonces como:

$$p(\boldsymbol{t} \mid O) \stackrel{\text{def}}{=} p(\boldsymbol{t} \mid \boldsymbol{z})$$

Definimos el reconocimiento de la expresión matemática como un problema de análisis estructural tal que el objetivo es obtener el árbol de análisis $\boldsymbol{t}$ más probable que tiene en cuenta los trazos de la secuencia de entrada $\boldsymbol{z}$.

Por tanto, la mejor estructura asociada con $\boldsymbol{z}$ se puede obtener de la forma:

$$\widehat{\boldsymbol{t}} \;=\; \underset{\boldsymbol{t}\in T}{\arg\max}\; p(\boldsymbol{t} \mid \boldsymbol{z}) \;=\; \underset{\boldsymbol{t}\in T}{\arg\max} \sum_{\substack{\boldsymbol{s}\in\mathcal{S} \\ \boldsymbol{s}=\text{yield}(\boldsymbol{t})}} p(\boldsymbol{t}, \boldsymbol{s} \mid \boldsymbol{z}),$$

Donde T representa el conjunto de todos los árboles de análisis posibles y $\boldsymbol{s} \in S$, una secuencias de símbolos compatible con t, donde $\boldsymbol{s} = \text{yield}(t)$ son las hojas del árbol de análisis t.

Si aproximamos la probabilidad como el árbol de análisis con mayor probabilidad, y asumimos que la parte estructural de la ecuación depende solo de la secuencia de pre-terminales s, la expresión se convierte en:

$$(\widehat{\boldsymbol{t}}, \widehat{\boldsymbol{s}}) \approx \underset{\substack{\boldsymbol{t}\in T, \boldsymbol{s}\in\mathcal{S} \\ \boldsymbol{s}=\text{yield}(\boldsymbol{t})}}{\arg\max}\, p(\boldsymbol{s} \mid \boldsymbol{z}) \cdot p(\boldsymbol{t} \mid \boldsymbol{s}) \tag{2.1}$$

donde $p(\boldsymbol{s}|\boldsymbol{z})$ representa la probabilidad de observación (símbolo) y $p(\boldsymbol{t}|\boldsymbol{s})$ representa la probabilidad estructural

Como discutimos, suelen usarse dos estrategias distintas para resolver este problema. La primera resuelve el problema en dos pasos. Primero calculando la segmentación de la entrada en símbolos matemáticos y segundo, calculando la estructura que relaciona todos los símbolos reconocidos [36]. La segunda resuelve el problema a través de una estrategia totalmente integrada para calcular Eq. (2.1) donde la segmentación de símbolos, reconocimiento y análisis estructural de la expresión de entrada son obtenidos globalmente [37]. En este trabajo buscamos definir un modelo capaz de utilizar la segunda estrategia.

Con éste objetivo en cuenta, el primer término se puede descomponer en:

$$p(\boldsymbol{s} \mid \boldsymbol{z}) \;=\; \prod_{i=1}^{N} p(s_i \mid z_i)$$

Donde, $p(s_i \mid z_i)$, es la probabilidad de que a la componente conexa, z_i, le corresponda el símbolo, s_i. Mientras que el segundo,

$$p(t \mid \boldsymbol{s}) = \frac{p(t, \boldsymbol{s})}{p(\boldsymbol{s})}$$

donde $p(\boldsymbol{s}) = \sum_{t'} p(t' \mid \boldsymbol{s})$, se puede calcular con el algoritmo *Inside*.

La búsqueda de la ecuación 2.1 se puede hacer por programación dinámica a partir de una versión del algoritmo de *Viterbi* para gramáticas bidimensionales.

En este trabajo proponemos una estrategia integrada para calcular la ecuación 2.1 donde la segmentación de símbolos, reconocimientos y análisis estructural de la expresión de entrada son determinados globalmente. De esta manera, se tiene en cuenta toda la información para obtener la expresión matemática más probable.

En la siguiente sección definimos el modelo de observación que calcula la probabilidad de reconocimiento y segmentación de símbolos $p(\boldsymbol{s}|\boldsymbol{z})$. La probabilidad que tiene en cuenta la estructura de la expresión matemática $p(t|\boldsymbol{s})$ se describe en la sección posterior.

2.2. Segmentación de símbolos

Como hemos visto, en el reconocimiento de expresiones matemáticas impresas la entrada se transforma a una secuencia de componentes conexas $\mathbf{z} = z_1 z_2 \ldots z_n$, que codifica una secuencia de pre-terminales de una gramática $\mathbf{s} = s_1 s_2 ... s_k, (1 \leq K \leq N)$ representados por símbolos matemáticos. Un símbolo es compuesto por uno o más trazos.

Antes de definir la estrategia de segmentación adoptada para modelar la probabilidad de símbolo, debemos introducir algunas definiciones formales preliminares.

Definición 2 *Dada una secuencia de N componentes conexas de entrada* $\mathbf{z}$*, y el conjunto conteniéndolos set*$(\mathbf{z}) = \{z_i | i : 1 \dots k\}$*, una segmentación de* $\mathbf{z}$ *en K segmentos es una partición del conjunto de trazos de entrada:*

$$b(\mathbf{z}, K) = \{b_i | i : 1...k\}$$

donde cada b_i *es un conjunto de componentes conexas representando una hipótesis de segmentación para un símbolo dado.*

Definición 3 *Definimos* B_k *como el conjunto de todas las posibles segmentaciones de los trazos de entrada* $\mathbf{z}$ *en K partes. De forma similar, definimos el conjunto de todas las segmentaciones B como:*

$$B = \bigcup_{1 \leq K \leq N} B_K$$

Una vez definido el conjunto de todas las posibles segmentaciones B, podemos calcular la probabilidad de la segmentación y reconocimiento de símbolos para una secuencia de componentes conexas dada $\mathbf{z}$. En la ecuación 2.1, podemos definir un modelo generativo $p(\mathbf{s}, \mathbf{z})$ en vez de $p(\mathbf{s}|\mathbf{z})$ dado que el término $p(\mathbf{z})$ no depende en las variables de maximización $\mathbf{s}$ y $\mathbf{t}$, podemos dejarlos fuera. El siguiente paso es remplazar la secuencia de N trazos de entrada $\mathbf{z}$ por su conjunto de segmentaciones previamente definido, $b = b(\mathbf{z}, K) \in B_K$ donde $1 \leq K \leq N$.

$$p(\mathbf{s}, \mathbf{z}) = \sum_{1 \leq K \leq N} \sum_{b \in B_K} p(\mathbf{s}, b)$$

Para desarrollar esta expresión, factorizamos con respecto al número de pre-terminales y asumimos las siguientes limitaciones: (i) Aproximamos el sumatorio usando maximización, (ii) La probabilidad de un posible segmento depende solo de las limitaciones espaciales de las componentes conexas de las cuales esta compuesto, (iii) la probabilidad de un símbolo

depende solo del conjunto de componentes conexas asociado, y (iv) El número de componentes conexas de un pre-terminal depende solo del símbolo que representa:

$$p(\mathbf{s}, \mathbf{z}) \approx \max_{K} \max_{b \in B_K} \prod_{i=1}^{K} p(b_i) p(s_i|b_i)$$

De esta ecuación podemos concluir que necesitamos definir dos modelos: un modelo de segmentación de símbolos, $p(b_i)$ y un modelo de clasificación de símbolos, $p(s_i|b_i)$. A continuación veremos estos modelos con más profundidad.

Muchos símbolos en notación matemática están compuestos por más de una componente conexa. Como hemos visto, en este trabajo proponemos un modelo donde la segmentación de componentes conexas esta basada en información espacial y geométrica y no en información temporal. También definimos B como el conjunto de todas las posibles segmentaciones. Dada la definición de B, es fácil ver que el tamaño es exponencial con el número de componentes conexas N. Ahora veremos como reducir efectivamente el número de segmentaciones a considerar y luego describimos el modelo de segmentación usado para calcular la probabilidad de cierta hipótesis $p(b_i)$.

Definición 4 *La distancia entre 2 componentes conexas z_i y z_j puede ser definida como la distancia euclidiana entre sus puntos más cercanos.*

Definición 5 *Una componente conexa z_i es considerada visible desde otra z_j si la linea recta entre los puntos más cercanos de ambas no cruza ninguna otra z_k. Si Una componente z_i no es visible desde z_j consideramos que la distancia entre ellas es infinita. Por ejemplo, dada la expresión de la Fig. 2.1, los trazos visibles desde z_4 son z_2 y z_3. Mas aun, sabemos que en un símbolo compuesto por múltiples componentes, éstas están cercanas. Por esta razón, solo consideramos hipótesis de segmentación b_i donde los trazos son cercanos entre si.*

Definición 6 *Una componente conexa z_i es considerada cercana a otra z_j si la distancia entre ellos es menor a un umbral dado. Usando estas definiciones, podemos caracterizar el conjunto de todas las hipótesis de segmentación posibles.*

Definición 7 *Sea G un grafo indirecto tal que cada trazo es un nodo y las aristas solo conectan componentes conexas que son visibles y cercanos. Entonces, una hipótesis de segmentación b_i es admisible si las componentes que contiene forman un sub-grafo conectado en G.*

Consecuentemente, una segmentación $\mathbf{b}(\mathbf{z}, K) = b_1, b_2, \ldots b_k$ es admisible si cada b_i es a su vez admisible. Éstas dos restricciones geométricas y espaciales reducen significativamente el número de posibles segmentaciones de símbolos.

2.3. Reconocedor de símbolos

Se han propuesto distintos enfoquen en la literatura para abordar el problema de la clasificación de símbolos, usando diferentes clasificadores: Redes Neuronales Artificiales[27], Support Vector Machines (SVM)[12], Gaussian Mixture Models (GMM)[25], elastic matching[17], Hidden Markov Models(HMMs) [33, 9] y Redes Neuronales Recurrentes (RNN)[38]. En este trabajo usaremos Redes Neuronales Convolucionales (CNN)[14] como clasificador ya que es capaz de obtener resultados competitivos en OCR utilizando solo información espacial de la entrada.

CNN son un modelo coneccionista basado en aplicar operaciones de convolución y pooling en capas, este modelo fue diseñado generar inherentemente independencia de traslaciones y de escalado sobre la entrada[14]. Ésta característica permite obtener buenos resultados de clasificación para tareas donde la misma estructura puede aparecer en varias posiciones, como la clasificación de símbolos matemáticos.

Para entrenar el modelo, definimos una lista de K clases de interés y generamos una colección de imágenes para cada clase utilizando la notación LaTeXdel símbolo para imprimir una imagen y manipulando la imagen para agregar variaciones aleatorias con respecto al tamaño, zoom, rotación y desplazamiento horizontal y vertical. Éstas imágenes son utilizadas para entrenar una CNN cuya capa de salida $\mathbf{z}$ tiene K neuronas y la probabilidad final se obtiene al aplicar:

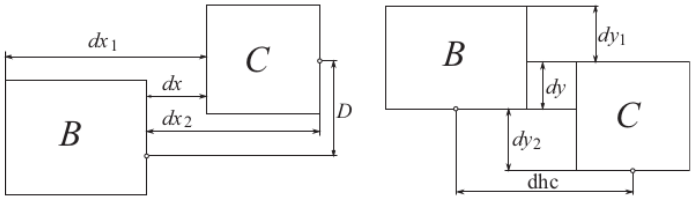
$$p(s_i \mid b_i) = \arg\max_j \frac{e^{z_j}}{\sum_{n=1}^{K} e^{z_n}}$$

para $j = 1, \ldots, K$ y $\mathbf{z} = (z_1, \ldots, z_K) \in R^K$

2.4. Relaciones espaciales

La definición de la ecuación para calcular la probabilidad estructural de una expresión matemática requiere un modelo de la relación espacial. Este modelo provee la probabilidad $p(r|BC)$ de que dos sub-problemas B y C estén relacionados acorde a la relación espacial r.

En este trabajo, abordamos el reconocimiento de expresiones matemáticas usando seis relaciones espaciales: derecha (BC), abajo $(\frac{B}{C})$, subscript (B_C), superscript (B^C), adentro $(\sqrt{C})$ y mroot $(\sqrt[C]{\ })$.



$$\mathrm{h}(B, C) = [H, D, \mathrm{dhc}, dx, dx_1, dx_2, dy, dy_1, dy_2]$$

Figura 2.2: Características geométricas para la clasificación de las relaciones espaciales entre B y C.

Para entrenar un clasificador estadístico, dadas dos regiones B y C, definimos nueve características geométricas basadas en sus bounding boxes [40] (ver Fig. 2.2). De esta manera, calculamos el vector de características $h(B, C)$ que representa su relación y puede ser usado para clasificación. Las características están definidas en la Fig. 2.2, donde H es la altura de la región C, D es la diferencia entre los centros verticales, y dhc es la diferencia entre los centros horizontales. Las características son normalizadas calculando la altura combinada de las regiones B y C.

Con vista a mejorar la colocación de centros verticales, dividimos los símbolos en cuatro categorías topográficas: ascendente (p.e. d o λ), descendente (p, μ), normal $(x, +)$, medio $(7, \Pi)$. Para símbolos normales el centroide es igual que el vertical. Para ascendentes se mueve para abajo a $(centroid + bottom)/2$. De igual manera los descendientes tienen el centro hacia arriba $(centroid + top)/2$. Finalmente, para los símbolos medios el centro vertical se define como $(top + bottom)/2$.

Una vez definido el vector de características representando una relación espacial, podemos entrenar un modelo de mixtura de gaussianas(GMM) usando muestras etiquetadas tal que la probabilidad del modelo de relaciones espaciales puede ser calculada como la probabilidad posterior provista por el GMM para la clase r:

$$p(r|BC) = P_{GMM}(r|h(B, C))$$

2.5. Modelo estructural

El modelo estadístico descrito en la Sección 2.1 define el problema de reconocer una expresión matemática como encontrar el árbol de análisis más probable t que de cuenta de las componentes conectadas de entrada $\mathbf{z}$. El problema es formalmente expresando en la ecuación 2.1 tal que se requieren dos probabilidades. En la sección previa calculamos la probabilidad del símbolo $p(\mathbf{s}|\mathbf{z})$. En esta sección definiremos la probabilidad estructural

$p(t|\mathbf{s})$.

Un modelo libre de contexto es un formalismo capaz de representar la estructura natural del lenguaje. Es un modelo apropiado para aplicar a notación matemática dadas las dependencias estructurales existentes entre diferentes elementos en una expresión. Usaremos una extensión bidimensional de las gramáticas incontextuales probabilísticas (GIP), un formalismo conocido y utilizado en reconocimiento de expresiones matemáticas [1, 5, 34, 2, 39].

Definición 8 *Una* Gramática Incontextual *(GI), G en una tupla de cuatro elementos* $(\mathcal{N}, \Sigma, S, \mathcal{P})$*,donde* $\mathcal{N}$ *es un conjunto finito de símbolos no-terminales,* Σ *es un conjunto finito de símbolos terminales* ($\mathcal{N} \cap \Sigma = \emptyset$)*,* $S \in \mathcal{N}$ *es el símbolo de inicio de la gramática, y* $\mathcal{P}$ *es un conjunto finito de reglas:* $A \rightarrow \alpha$, $A \in \mathcal{N}$, $\alpha \in (\mathcal{N} \cup \Sigma)^+$.

Una GI en Notación Formal de Chomsky es una GI en donde las reglas son de la forma $A \rightarrow BC$ o $A \rightarrow a$, donde $A, B, C \in \mathcal{N}$ y $a \in \Sigma$.

Definición 9 *Una* GI Probabilista *(GIP) es definida como un par* (G, p)*, donde G es una GI y* $p : P \rightarrow]0, 1]$ *es una función de probabilidad de la aplicación de reglas tal que* $\forall A \in \mathcal{N} : \sum_{i=1}^{n_A} p(A \rightarrow \alpha_i) = 1$*, donde* n_A *es el número de reglas asociadas con el símbolo no-terminal A.*

Definición 10 *Una* GIP Bi-dimensional *(GIP-2D),* $\mathcal{G}$*, es una generalización de una GIP, donde símbolos terminales y no-terminales describen regiones bidimensionales. Esta gramática en CNF resulta en dos tipos de reglas: reglas terminales y reglas binarias.*

Primero, las reglas terminales $A \rightarrow a$ representan los símbolos matemáticos que son finalmente los símbolos terminales de la GIP-2D. Segundo, las reglas binarias $A \xrightarrow{r} BC$ tienen un parámetro adicional, r, que representan

una relación espacial dada, y su interpretación es que las regiones B y C deben estar arregladas especialmente de acuerdo a la relación espacial r.

El modelo GIP-2D nos permite calcular la probabilidad estructural de una expresión matemática en términos de la probabilidad conjunta $p(t, \mathbf{s})$ tal que en CNF es calculada como:

$$p(t, \mathbf{s}) = \prod_{(A \to a, t)} p(a|A) \prod_{(A \to BC, t)} p(BC|A)$$

donde $p(a|A)$ es la probabilidad de la regla $A \to a$ y representa la probabilidad de que a es derivado desde A. $(A \to a, t)$ denota todas las reglas $(A \to a)$ contenidas en el árbol de análisis t. En la expansión de 2D definida sobre GIP, la composición de sub-problemas tiene una restricción adicional basada en la relación espacial r. Sea la relación espacial r entre dos regiones una variable oculta, la probabilidad de una regla binaria se escribe como:

$$p(BC|A) = \sum_{r} p(BC, r|A)$$

Cuando la probabilidad interior en la suma previa es estimada desde muestras, el modo es el término dominante. Por lo tanto, aproximando sumas con maximizaciones, y asumiendo que la probabilidad de una relación espacial depende solo de los sub-problemas B y C involucrados, la probabilidad estructural de la expresión matemática se convierte en:

$$p(t, \mathbf{s}) \approx \prod_{(A \to a, t)} p(a|A) \prod_{(A \to BC, t)} \max_{r} p(BC|A) p(r|BC)$$

donde $p(a|A)$ y $p(BC|A)$ son las probabilidades de las reglas de la gramática, y $p(r|BC)$ es la probabilidad de que regiones codificadas por no terminales B y C están relacionadas acorde a la relación espacial r.

2.5.1. Algoritmo de análisis para GIP-2D

El cálculo de los N-mejores árboles de análisis desde GIP-2D para expresiones matemáticas ha sido diseñado como un proceso de 2 pasos. En

este primer paso buscaremos obtener el mejor árbol de análisis para la EM de entrada y en un segundo paso, aprovechando las estructuras creadas, calcularemos los $(N-1)$ mejores árboles restantes. Siguiendo el algoritmo de análisis para REM online desde GIP-2D presentado en [37], proponemos una modificación de este para calcular los mejores árboles de análisis en base a las siguientes ecuaciones recursivas, y agregamos definiciones teóricas de las estructuras que nos permitirán almacenar la información necesaria entre los pasos a ejecutar.

Definición 11 *Sea $\mathcal{G}$ una GIP-2D, y sea $\boldsymbol{z}$ un conjunto de componentes conectadas. Primero, definimos $\mathcal{T}(A,a)$ como el conjunto de árboles cuya raíz es (A,a), donde $A \in \mathcal{N}$ es un símbolo no-terminal y $a \in \wp(\boldsymbol{z})$ es un cierto rango de la entrada. $T(A,a) \in \mathcal{T}(A,a)$, representa uno de los posibles árboles en $\mathcal{T}(A,a)$. Segundo, definimos una función probabilista $p : \mathcal{T} \to [0,1]$, como sigue:*

***(I)** Si hay una regla terminal $A \to s$ y un conjunto con una sola componente conectada, $\{z_i\}, 1 \leq i \leq |\boldsymbol{z}|$, entonces tenemos en árbol $\langle A, \{z_i\}\rangle \in \mathcal{T}(A,\{z_i\})$, y su probabilidad es:*

$$\begin{aligned} p(\langle A,\{z_i\}\rangle) &= \sum_{s\in\Sigma} p(s \mid A)\ p(\{z_i\} \mid s) \\ &\approx \frac{1}{|\boldsymbol{z}|}\ \max_s \left\{ \frac{p(s \mid A)\ p(s \mid \{z_i\})}{p(s)} \right\} \end{aligned} \tag{2.2}$$

donde $p(s|A)$ es la probabilidad de la regla terminal, $p(s|\{z_i\})$ es la probabilidad propuesta por el clasificador de símbolos, y $p(s)$ es la probabilidad a priori de los símbolos. Acá consideramos todas las componentes conectadas equiprobables.

***(II)** Si hay una regla binaria $A \to BC$, un árbol $T_1(B,b)$ en $\mathcal{T}(B,b)$ y un árbol $T_2(C,c)$ en $\mathcal{T}(C,c)$, tal que $b \cap c = \emptyset$ y $a = b \cup c$, entonces nos referimos al árbol $\langle T(A,a), T_1, T_2\rangle$ como el árbol con raíz A, sub-árbol izquierdo T_1, y sub-*

árbol derecho T_2, y su probabilidad es aproximada como:

$$\begin{aligned} & p(\langle (A,a), T_1, T_2 \rangle) \\ & \quad \approx \quad \max_r \; p(r \mid BC)\; p(BC \mid A)\; p(T_1)\; p(T_2) \end{aligned} \tag{2.3}$$

donde $p(BC|A)$ es la probabilidad de la regla binaria, y $p(r|BC)$ es la probabilidad que una región codificada por los no-terminales B y C estén arregladas de acuerdo a la relación especial r.

Ahora definimos $\gamma^1(A,a)$ como la probabilidad de que $A \in \mathcal{N}$ sea la 1-Mejor solución del conjunto de componentes conectadas a.

Inicialización. De acuerdo a la Eq. (2.2) y dado que el clasificador de símbolos puede proveer varias alternativas de reconocimiento, con diferentes simbolos no-terminales $A, A \in \mathcal{N}$, para cada componente conectada $\{z_i\}, 1 \leq i \leq |\boldsymbol{z}|$, definimos,

$$\begin{aligned} \gamma^1(A,\{z_i\}) &= \max_{T \in \mathcal{T}^1(A,\{z_i\})} p(T) \\ T^1(A,\{z_i\}) &= \arg\max_{T \in \mathcal{T}^1(A,\{z_i\})} p(T) \end{aligned}$$

El conjunto de árboles $\mathcal{T}^1(A,\{z_i\})$ cuyas raíces son de la forma $(A,\{z_i\})$ pueden definirse como

$$\mathcal{T}^1(A,\{z_i\}) \;=\; \{\, \langle A,\{z_i\}\rangle : A \to s;\; p(s \mid \{z_i\}) > 0 \,\} \tag{2.4}$$

Recursión. De acuerdo a la Eq. (2.3), $\forall A \in \mathcal{N}$ y $\forall a \in \wp(\boldsymbol{z})$, tal que $|a| > 1$:

$$\begin{aligned} \gamma^1(A,a) &= \max_{T \in \mathcal{T}^1(A,a)} p(T) \\ T^1(A,a) &= \arg\max_{T \in \mathcal{T}^1(A,a)} p(T) \end{aligned} \tag{2.5}$$

donde, para $a \in \wp(\mathbf{z})$

$$\begin{aligned} \mathcal{T}^1(A,a) \;=\; \{\, \langle (A,a), T^1(B,b), T^1(C,c)\rangle : \\ A \to BC;\; b \cap c = \emptyset;\; a = b \cup c \,\} \end{aligned} \tag{2.6}$$

denota el conjunto de árboles candidatos con raíz A que utiliza las componentes conexas a.

Abordamos el cómputo de la caja de inclusión máxima asociada a a, de talla $I_a \times J_a$, y definida como $[(x_a, y_a) \times (x_a + I_a, y_a + J_a)]$, a partir de las cajas de inclusión máxima de ambos subproblemas.

$$\begin{cases} x_a = \min\{x_b, x_c\} \\ y_a = \min\{y_b, y_c\} \\ I_a = \max\{x_b + I_b, x_c + I_c\} - x_z + 1 \\ J_a = \max\{y_b + J_b, y_c + J_c\} - y_z + 1 \end{cases}$$

Finalmente, la hipótesis más probable y su árbol de derivación asociado $\widehat{t}$ que da cuenta de la expresión de entrada puede obtenerse en

$$\gamma_N^{\mathbf{z}}[S] \;=\; p(S \stackrel{*}{\Rightarrow} \{z_1, \ldots z_N\})$$

donde $\mathbf{z}$ es el conjunto de las N componentes conexas extraídas de la imagen O y S es el axioma de la gramática bidimensional. Esta es la probabilidad del mejor árbol de derivación, t, y sus hojas, la mejor secuencia de símbolos, $\mathbf{s}$.

El problema de calcular el mejor árbol de derivación para O consiste en resolver las ecuaciones Eq. (2.2)yEq. (2.3) para encontrar $\gamma_N^{\mathbf{z}}[S]$. El algoritmo CYK (Fig. 2.3) es un algoritmo de programación dinámica que calcula $\gamma_N^{\mathbf{z}}[S]$ iterativamente para valores incrementales de la longitud $l = k - i$, garantizando que la parte derecha de la ecuación ha sido previamente calculada cuando ha de usarse.

```
Algoritmo CYK para GIP−2D.
Calcular γ_l^a[A]^1 para todo A ∈ V usando el algoritmo CYK.
Input: GIP−2D G_s = (N, T, P, S, Pr, spr) in CNF
  and z = {z_1, z_2, ..., z_n} ∈ T*
Output: γ_l^a[A]^1 probabilidad del arbol de derivacion
  mas probable

for i = 1...n do
  for (A → z_i) ∈ P if p(A → z_i) > 0,0 do
    t[1] = t[1] ∪ (A, z_i, p(A → z_i))
    γ_1^{z_i}[A] = max_s p(s|A) · p(s|{z_i})

for j = 2...n do
  for a = 1...j−1 do
    for c1 = (B, r_1, p_B) ∈ t[a] do
      for c2 = (C, r_2, p_C) ∈ t[j−a] do
        for (A →^spr BC) ∈ P do
          prob = p_B · p_C · p(A →^spr BC)
          if prob > 0.0 then
            t[j] = t[j] ⊎(A, r_1 ⊕ r_2, prob)

return (S, z, p) ∈ t[n]
```

Figura 2.3: Algoritmo CYK para GIP-2D. (A, r, p) representa que el no-terminal A describe la región r con probabilidad p.

2.5.2. Complejidad y espacio de búsqueda

Hemos definido un enfoque integrado para reconocimiento de expresiones matemáticas basado en el análisis de GIP-2D. El algoritmo de programación dinámica es definido por las ecuaciones recursivas correspon-

dientes. El paso de inicialización es realizado por la ecuación (2.4), mientras que el caso general se calcula en base a la ecuación 2.6. Ademas de la definición formal, existen algunos detalles de el espacio de búsqueda de el algoritmo de análisis que necesitan mayor explicación.

Una vez varias hipótesis de símbolos han sido creadas durante la inicialización, el caso general es el núcleo del algoritmo donde hipótesis de tamaño cada vez mayor $2 \leq l \leq N$ son generadas. Para un tamaño dado l, tenemos que probar los distintos tamaños para cortar l en b_B y b_C tal que $l = l_B + l_C$. Una vez obtenidos, para cada conjunto de componentes conexas b_B tenemos que probar toda posible combinación con otro conjunto b_C usando las reglas binarias de la gramática $A \xrightarrow{r} BC$. De acuerdo a ésto, podemos ver que la complejidad temporal de analizar una expresión de entrada de N componentes conexas es $O(N^4|P|)$ donde $|P|$ es el número de producciones de la gramática. Sin embargo, la complejidad puede ser reducida restringiendo el espacio de búsqueda.

La intuición es que, dado un conjunto de componentes conexas b_B, no es necesario tratar de combinarlo con todo otro conjunto b_C. Un conjunto de componentes b_B define una región en el espacio, permitiéndonos limitar el conjunto de hipótesis b_C a aquellas que caen en una región de interés. Aplicamos esta intuición de la siguiente manera. Dado un trazo z_i definimos su región asociada $r(z_i) = (x, y, s, t)$ en el espacio bidimensional como el bounding box mínimos que contiene ese trazo, donde (x, y) es la coordenada de arriba-izquierda y (s,t) la de abajo-derecha. De igual manera, dado un conjunto de componentes conexas $b = \{z_j | 1 \leq j \leq |b|\}$ definimos $r(b) = (x_b, y_b, s_b, t_b)$ como el mínimo rectángulo que contiene todos las componentes conexas $z_j \in b$. Por lo tanto dada una región espacial $r(b_B)$ obtenemos solo la hipótesis b_C cuya región $r(b_C)$ cae dentro de un área $\mathcal{R}$ relativa a b_B.

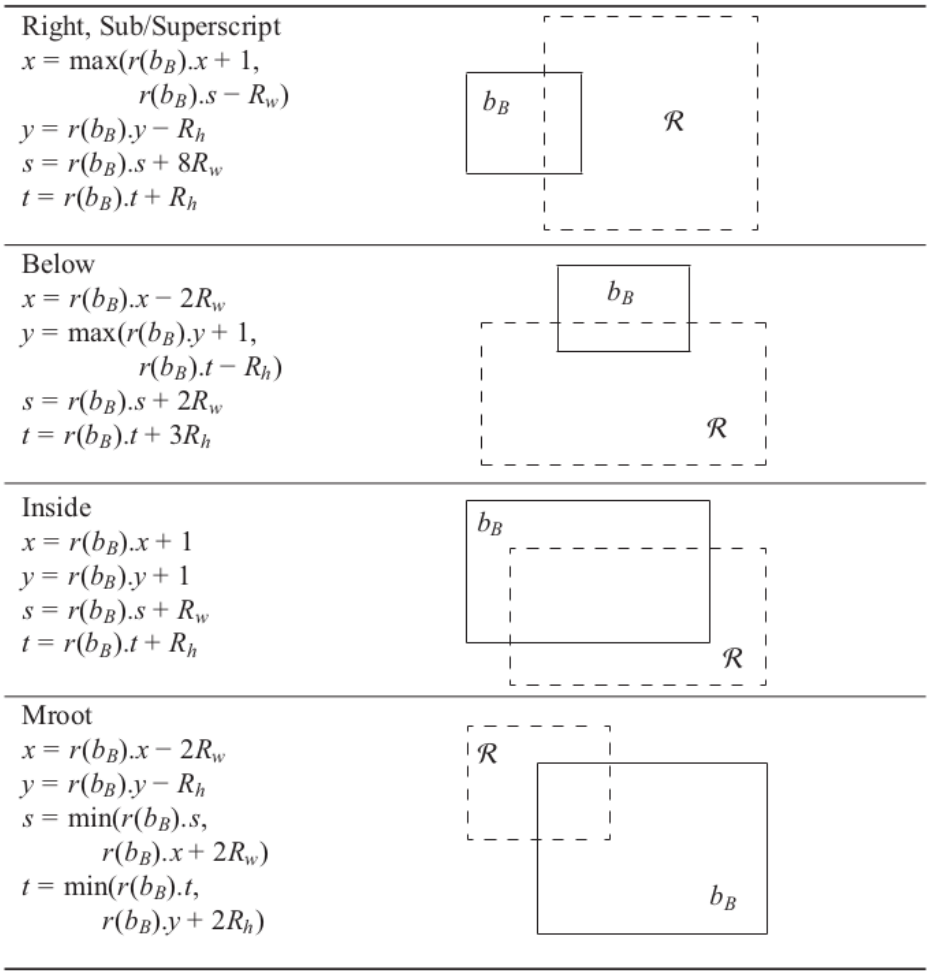


Figura 2.4: Regiones espaciales definidas para recuperar hipótesis relativas a b_B de acuerdo a diferentes relaciones.

La Fig. 2.4 muestra las definiciones de las regiones en el espacio donde se considera relevante obtener hipótesis para combinar con b_B. Las dimensiones de tamaños normalizados del símbolo (R_w, R_h) se calculan de la siguiente manera: R_w el máximo entre la altura media y mediana del trazo de entrada. Estos cálculos son independientes de la resolución de la en-

trada. El tamaño normalizado del símbolo también se usa para normalizar otras medicas relacionadas a la distancia en el modelo, como determinar que componentes están más cercanas o la normalización de características en el modelo de segmentación.

Para poder obtener eficientemente las hipótesis que caen en una región dada $\mathcal{R}$, cada vez que un conjunto de hipótesis de tamaño l_A es calculado, ordenamos el conjunto de acuerdo a la coordenada x de cada región $r(b_A)$ asociada con $\gamma(A, b_A, l_A)$. Esta operación de ordenado tiene un coste de $O(N \log N)$. Después, dado un rectángulo $r(b_B)$ en el espacio de búsqueda y un tamaño l_C, podemos recuperar la hipótesis $\gamma(C, b_C, l_C)$ que cae en esa área realizando una búsqueda binaria sobre el conjunto en $O(\log N)$. Aunque las regiones están relacionadas en dos dimensiones y ordenadas en torno a una, este enfoque es razonable ya que las expresiones matemáticas crecen principalmente de izquierda a derecha

Asumiendo que esta búsqueda binaria recupera un número bajo de hipótesis, la complejidad final obtenida es $O(N^3 \log N|P|)$. Ademas, muchas hipótesis improbables son podadas durante el proceso de análisis.

Capítulo 3

N-mejores e Hipergrafos para GIP-2D

En el capítulo anterior hemos visto los modelos y el proceso utilizado para obtener el árbol de derivación más probable $\widehat{t}$ dada la imagen de una expresión matemática O usando un clasificador de simbolos y un modelo de GIP-2D. Sin embargo, como remarcamos en el capítulo de introducción, un framework para la indexación de expresiones matemáticas no debería considerar que el resultado que obtiene es necesariamente correcto y además, un modelo de REM debería ser capaz de lidiar con la ambigüedad innata presente en ecuaciones.

Para resolver estos problemas e inspirados por las estrategias usadas en el proyecto HIMANIS, en este trabajo proponemos utilizar hipergrafos para modelar las expresiones y sub-expresiones que describen la probabilidad de una secuencia de entrada. Para obtener los hipergrafos definimos una generalización de la ecuación 2.1 para obtener los N mejores árboles de análisis de una expresión de entrada y después usamos los árboles generados para construir la estructura de hipergrafo y normalizarla.

3.1. N-mejores derivaciones para GIP-2D

Basándonos en el framework matemático propuesto en el capítulo anterior para obtener el árbol de derivación más probable para describir una expresión matemática de entrada, junto con símbolos y componentes conectadas usados en la expresión $(\widehat{t}, \widehat{s})$. En esta sección usamos el algoritmo propuesto en [10] para calcular los N mejores mejores árboles en GIP y vamos a extender el modelo GIP-2D propuesto para poder obtener las N mejores estructuras posibles cada una con el conjunto de símbolos y componentes que utiliza.

En este segundo paso del proceso se aprovechan las estructuras creadas durante el cálculo del mejor árbol de análisis (ver Eq. (2.4) y (2.6)). Para el cálculo del N-mejor árbol de análisis, seguimos [11] para calcular la enumeración recursiva de los mejores árboles de análisis, que aprovecha el orden parcial entre diferentes árboles candidatos. A continuación, generalizaremos los algoritmos recursivos dados en la sección previa para calcular los N-mejores árboles, y después propondremos un algoritmo para resolver las ecuaciones generalizadas.

El algoritmo propuesto en [10] puede modificarse para ser utilizado nuestro modelo para obtener los mejores árboles de derivación de la gramática junto con los símbolos más probables en cada solución, en particular, vamos a considerar como calcular $\gamma_l^a[A]^n$ para $1 \leq n \leq N$, y obtener $\gamma_N^{\mathbf{z}}[S]^n$ como solución al problema. Extenderemos el modelo GIP-2D presentado en el capítulo anterior para calcular los N-mejores árboles de derivación y luego definiremos un algoritmo para resolver las extensiones realizadas.

3.1.1. Algoritmo de los N-mejores análisis

Si estudiamos que expresiones deberían ser consideradas candidatos para $\gamma_l^a[A]^n$. Si $l = 1$ la existencia de $\gamma_l^a[A]^n$ para $n > 1$ depende de si

existe ambigüedad en el clasificador de símbolos para la componente conexa a. para el caso donde $l > 1$ es claro que para calcular $\gamma_l^a[A]^n$ no es necesario considerar árboles de la forma $\langle A, \gamma_{|b|}^b[B]^p, \gamma_{|c|}^c[C]^q \rangle$ con $p > n$ o $q > n$ porque hay al menos n árboles entre ellos con menor o igual peso [11]. Por lo tanto $\gamma_l^a[A]^n$ puede escogerse como el mejor árbol diferente de $\gamma_l^a[A]^1, \ldots, \gamma_l^a[A]^{n-1}$ en el conjunto

$$\langle A, \gamma_{|b|}^b[B]^p, \gamma_{|c|}^c[C]^q \rangle : A \xrightarrow{r} BC \in P, 1 \leq p \leq n, 1 \leq q \leq n; \\ a = b \cup c, b \cap c = \emptyset, \mathrm{y} |a| = |b| + |c|$$

Pero podemos hacer mejor que calcular los N mejores árboles para cada posible raíz $\gamma_N^{\mathbf{z}}[S]^n$ dado que existe un orden parcial entre algunos elementos del conjunto de árboles. Basados en las relaciones (Representadas esquemáticamente en la Fig 3.1)

$$\gamma_{|b|}^b[B]^p \cdot \gamma_{|c|}^c[C]^1 \geq \gamma_{|b|}^b[B]^{p+1} \cdot \gamma_{|c|}^c[C]^1$$

$$\gamma_{|b|}^b[B]^p \cdot \gamma_{|c|}^c[C]^q \geq \gamma_{|b|}^b[B]^p \cdot \gamma_{|c|}^c[C]^{q+1}$$

podemos definir un conjunto más pequeño $\tau_l^a[A]^n$ de árboles con raíz A de tal forma que conservamos la garantía de que $\gamma_l^a[A]^n$ es el mejor entre ellos.

$$
\begin{array}{ccccc}
\langle A, \gamma^b_{|b|}[B]^1, \gamma^c_{|c|}[C]^1\rangle & \Rightarrow & \langle A, \gamma^b_{|b|}[B]^1, \gamma^c_{|c|}[C]^2\rangle & \Rightarrow & \langle A, \gamma^b_{|b|}[B]^1, \gamma^c_{|c|}[C]^3\rangle \Rightarrow \ldots \\
\Downarrow & & & & \\
\langle A, \gamma^b_{|b|}[B]^2, \gamma^c_{|c|}[C]^1\rangle & \Rightarrow & \langle A, \gamma^b_{|b|}[B]^2, \gamma^c_{|c|}[C]^2\rangle & \Rightarrow & \langle A, \gamma^b_{|b|}[B]^2, \gamma^c_{|c|}[C]^3\rangle \Rightarrow \ldots \\
\Downarrow & & & & \\
\langle A, \gamma^b_{|b|}[B]^3, \gamma^c_{|c|}[C]^1\rangle & \Rightarrow & \langle A, \gamma^b_{|b|}[B]^3, \gamma^c_{|c|}[C]^2\rangle & \Rightarrow & \langle A, \gamma^b_{|b|}[B]^3, \gamma^c_{|c|}[C]^3\rangle \Rightarrow \ldots \\
\Downarrow & & & & \\
\ldots & & & &
\end{array}
$$

Figura 3.1: Representación esquemática del orden parcial entre algunos árboles candidatos

Como mencionamos arriba, el problema es obtener $T^n(S, \boldsymbol{z})$ para $1 \leq n \leq N$. Para ésto, tenemos que definir $T^n(A, a),\ 1 \leq n \leq N$, $A \in \mathcal{N}$ y $a \in \wp(\boldsymbol{z})$, como el n mejor árbol entre aquellos que tienen raíz A y el conjunto de componentes conectadas a. Por lo tanto, $T^n(A, a)$ puede escogerse como el mejor árbol de análisis desde $T^1(A, a), \ldots, T^{n-1}(A, a)$ en el conjunto $\mathcal{T}^n(A, a)$, que esta definido abajo. Análogamente al caso del 1-mejor (ver Eq. (2.4)) y teniendo en cuenta Eq. (2.2), podemos definir $\mathcal{T}^n(A, \{z_i\})$ para cada $A \in \mathcal{N}$, $1 \leq i \leq |\boldsymbol{z}|$, y $1 < n \leq N$, como,

$$
\mathcal{T}^n(A, \{z_i\}) \ = \ \mathcal{T}^{n-1}(A, \{z_i\}) - \{T^{n-1}(A, \{z_i\})\} \tag{3.1}
$$

Además, para $|a| > 1$ y considerando Eq. (2.6), definimos $\mathcal{T}^n(A, a)$ como,

$$
\begin{aligned}
\mathcal{T}^n(A, a) = & \ (\mathcal{T}^{n-1}(A, a) - \{T^{n-1}(A, a)\}) \\
& \cup \ \{\langle T(A, a), T^{p+1}(B, b), T^q(C, c)\rangle\} \\
& \cup \ \{\langle T(A, a), T^p(B, b), T^{q+1}(C, c)\rangle\}
\end{aligned} \tag{3.2}
$$

donde $A \to BC;\ a, b, c \in \wp(\boldsymbol{z});\ b \cap c = \emptyset;\ a = b \cup c$ y $1 \leq p, q \leq n$. Los valores p y q son usados para seguir soluciones previas que han sido usadas.

Esto es, si $T^p(B,b)$ ha sido usado en un árbol previo, entonces $T^{p+1}(B,b)$ es necesario y se combina con $T^q(C,c)$.

Siguiendo [11] y asumiendo que $\{\langle T(A,a), T_1, T_2\rangle\}$ describe el conjunto vacío si T_1 o T_2 no existen. Entonces tenemos que:

$$\begin{aligned} \gamma^n(A,a)^n &= \max_{T\in\mathcal{T}^n(A,a)} p(T) \\ T^n(A,a) &= \arg\max_{T\in\mathcal{T}^n(A,a)} p(T) \end{aligned} \tag{3.3}$$

Por lo tanto, el problema de calcular el N-mejor árbol de análisis para una imagen consiste en usar las ecuaciones (2.4) y (2.5) para obtener $T^1(S,\boldsymbol{z})$ y, tomando ventaja de los cálculos previos, usar las ecuaciones (3.1) y (3.2) para encontrar $T^2(S,\boldsymbol{z}),\ldots,T^N(S,\boldsymbol{z})$. El algoritmo que se muestra en la figura 3.3 resuelve las ecuaciones para valores incrementales de n, recursivamente empezando desde el árbol $T^n(S,\boldsymbol{z})$. El algoritmo hace uso de los procedimientos recursivos `NextTree()`, para $n > 1$, y una vez $T^n(S,\boldsymbol{z}) =$ `NextTree`$(T^{n-1}(S,\boldsymbol{z}),n)$ esta disponible, `NextTree`$(\langle T(A,a), T^p(B,b), T^q(C,c)\rangle, n)$ calcula $T^n(S,\boldsymbol{z})$ de acuerdo a la ecuación (3.2).

Esto requiere dos nuevos árboles candidatos $T^{p+1}(B,b)$ y $T^{q+1}(C,c)$, si alguno de estos candidatos no ha sido calculado antes, llamamos a la función `NexTree` recursivamente sobre él. Un ejemplo de las soluciones producidas puede verse en la figura 3.2.

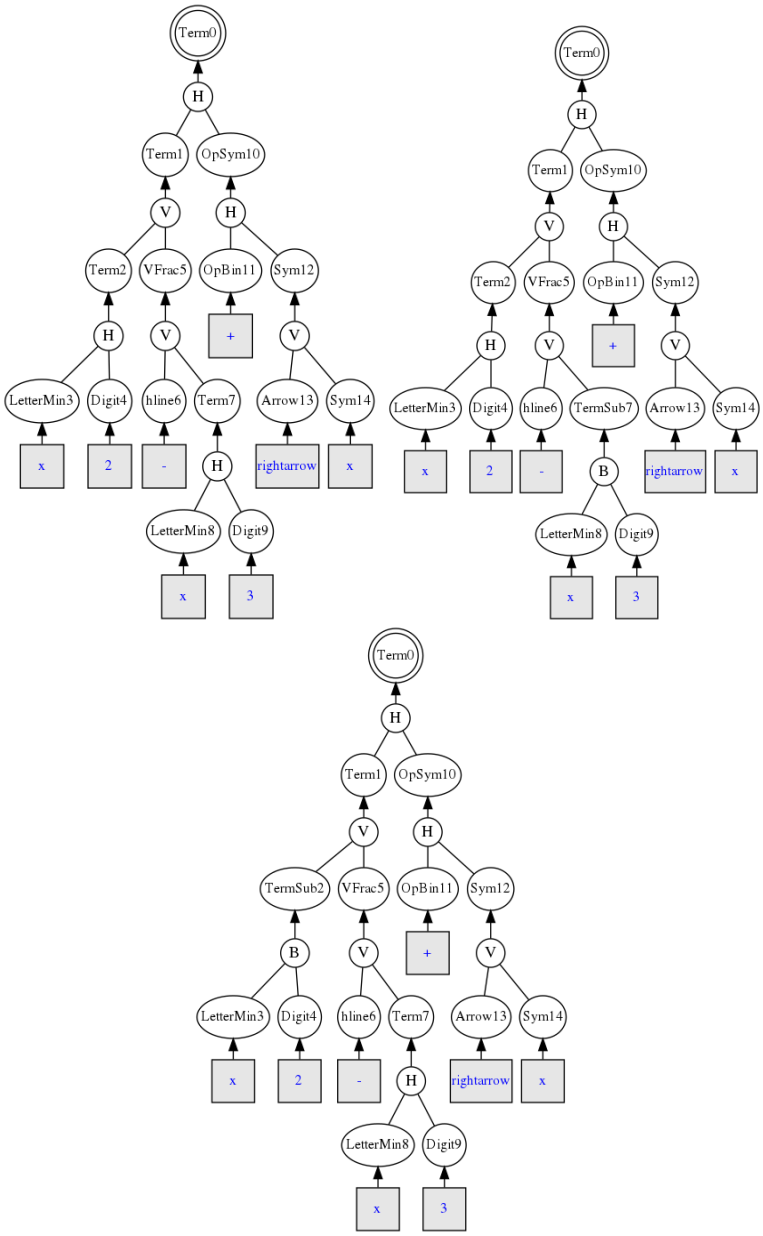


Figura 3.2: 3 mejores derivaciones para la expresión "$\frac{x_2}{x_3} + \vec{x}$". Arriba a la izquierda la 1-Mejor, arriba a la derecha la 2-Mejor y abajo la 3-Mejor

El algoritmo CYK-2D corre en tiempo $O(|P||x|^3 \log |x|)$. El número de conjuntos diferentes $\mathcal{T}(A, a)$ es $O(|a|^2|\Sigma|)$ y, en el peor caso, todos deben ser inicializados en tiempo linear con respecto al tamaño del conjunto. El cálculo de los N mejores árboles de análisis requiere como mucho $N|x|$ llamadas a *NextTree*. El tiempo total requerido por el algoritmo completo para calcular los N mejores árboles es $O(|P||x|^3 \log |x| + N|x| \log(|x| + N))$. Este análisis de complejidad computacional esta basado en asunciones del peor caso posible que podrían ser muy pesimistas. En la práctica, puede esperarse que incluso para valores altos de N, no todos los conjuntos de candidatos son inicializados y el número de llamadas recursivas seria bastante menor que $N|x|$.

```
T^1(S,z) = CYK(z)
for n=2 to N:
  T^n(S,z) = NextTree(T^{n-1}(S,z), n)
return {T^1(S,z), T^2(S,z), ..., T^N(S,z)}

def NextTree(⟨T(A,a), T^p(B,b), T^q(C,c)⟩, n):
  T^n(A,a) = T^{n-1}(A,a) − {T^{n-1}(A,a)}
  if |a| > 1:
    if T^{p+1}(B,b) not set:
      NextTree(T^p(B,b), p+1)
    if T^{p+1}(B,b) != None:
      p = p(BC | A) · p(r | BC) · γ^{p+1}(B,b) · γ^q(C,c)
      if p > 0,0:
        T^n(A,a) = T^n(A,a) ∪
            {⟨T(A,a), T^{p+1}(B,b), T^q(C,c)⟩}
    if T^{q+1}(C,c) not set:
      NextTree(T^q(C,c), q+1)
    if T^{q+1}(C,c) != None:
      p = p(BC | A) · p(r | BC) · γ^p(B,b) · γ^{q+1}(C,c)
      if p > 0,0:
        T^n(A,a) = T^n(A,a) ∪
            {⟨T(A,a), T^p(B,b), T^{q+1}(C,c)⟩}

  if T^n(A,a) != None:
    γ^n(A,a) = máx_{T∈T^n(A,a)} p(T)
    T^n(A,a) = arg max_{T∈T^n(A,a)} p(T)
  else:
    T^n(A,a) = None
```

Figura 3.3: Algoritmo para generar los N-mejores árboles de análisis para una expresión de entrada.

3.1.2. Parametrización de GIP-2D

Utilizando el método propuesto en la sección anterior podemos generar las N estructuras más probables que describen una expresión de entrada, obtener múltiples hipótesis nos permite empezar a afrontar los problemas de ambigüedad en las expresiones matemáticas. Intuitivamente, si una expresión matemática puede ser descrita por múltiples árboles de derivación, todas las estructuras y símbolos que describan la formula van a aparecer en las N-mejores dado un número de N suficientemente grande. Aprovechando esta característica presentaremos una estrategia para utilizar los árboles de derivación presentes en las N-mejores para aproximar los parámetros óptimos de una GIP-2D.

Para obtener una buena aproximación necesitamos un dataset de expresiones matemáticas escritas en LaTeXde donde extraer las probabilidades reales, usamos el lenguaje LATEX para describir las expresiones dado a su extensivo uso en la comunidad científica, lo que facilita la obtención de un dataset público; a que es posible imprimir cualquier expresión LaTeXen una imagen; y a que podemos hacer una simple traducción de los árboles de derivación generados por nuestro modelo a esa notación. Gracias a esto podemos escribir un programa que dada una entrada LaTeXdel dataset, imprima la imagen correspondiente y usando valores iniciales en la gramática, generar los mejores árboles de derivación e imprimir su traducción LaTeXen otra imagen. Usando una medida de distancia entre la imagen original (ground truth) y la generada con un árbol de derivación es posible detectar las soluciones verdaderas generadas, y, para cada árbol verdadero, aumentar el peso de las reglas que se aplican en cada sub-estructura, mientras se reduce el de reglas aplicadas en soluciones erróneas. Al final del proceso cuando se han recorrido todos los ejemplos del dataset de entrada, solo es necesario normalizar las reglas de la GIP-2D.

```
I # Numero de iteraciones
scores # Tabla de reglas con probabilidad
alpha = 0.10 # Factor de aprendizaje
epsilon = 0.35 # Umbral para aciertos

step = 0
while step < I:
  for x in training:
    imgx = renderizar(x)
    #Obtengo los n-mejores arboles
    nbestx = n-best(imgx)
    #sumo la probabilidad de todos los arboles
    ptotal = sum(nbestx)
    for tree in nbestx:
      imgn = renderizar(tree)
      dist = distance(imgx, imgn)
      prob = p(tree) # probabilidad de la n-solucion
      if dist < epsilon:
        for regla in tree:
          scores[regla] += prob/pTotal
      for regla in tree:
        scores[regla] -= alpha * prob/pTotal
  normalizar(scores)
  step++
```

Figura 3.4: Algoritmo de parametrización de GIP-2D

Usando este proceso podemos centrarnos en especificar una gramática capaz de combinar debidamente los símbolos de expresiones matemáticas y utilizar una versión inicial de los parámetros junto con un dataset de expresiones matemáticas en LaTeXpara ajustar los parámetros a una aproximación de las probabilidades verdaderas.

3.2. Hipergrafos

A partir del análisis de las n-mejores derivaciones para gramáticas bidimensionales descrito en la sección anterior se puede generar un hipergrafo asociado, combinando todas las reglas y símbolos utilizados.

Un hipergrafo es una generalización del concepto de grafo, donde los ejes (ahora llamados hiperejes) pueden conectar varios nodos al mismo tiempo. Usamos hipergrafos como una representación del resultado de las N-mejores derivaciones con una GIP-2D dada. Dado un árbol determinado, los no-terminales representan los nodos y las reglas representar los hiperejes.

Pretendemos usar hipergrafos para representar los n-mejores árboles de análisis para una GIP-2D. Nodos e hiperejes pueden estar compartidos entre diferentes árboles de análisis si representan la misma información. Por lo que, los hipergrafos proveen una representación compacta del espacio de análisis. Como veremos abajo, esta representación compacta nos permite desarrollar algoritmos de inferencia eficientes.

3.2.1. Notación de hipergrafos

Primero, presentamos unas definiciones preliminares acerca de hipergrafos. Para información más detallada, véase [8].

Definición 12 *Un hipergrafo dirigido por pero $\mathcal{H}$ es un par $(\mathcal{V}, \mathcal{E})$ donde $\mathcal{V}$ es un conjunto de nodos (o vértices) y $\mathcal{E}$ es un conjunto de hiperarcos dirigidos (o hiperejes).*

Como queremos usar el hipergrafos como una representación compacta de los n-mejores árboles de análisis para una GIP-2D, $\mathcal{G} = (\mathcal{N}, \Sigma, S, \mathcal{P})$, redefinimos las nociones de nodo e hiperarco.

Definición 13 *Un nodo $v \in \mathcal{V}$ es un par $(n(v), s(v))$ donde $n(v)$ es el tag de nodo y $s(v)$ es el span (o información posicional) asociado con este nodo.*

En nuestro caso, el span, $s(v) = a$, representa el conjunto de componentes conectadas asociadas al tag de nodo, $n(v)$. En la otra mano, si $n(v) = A \in \Sigma$ (simbolo terminal) entonces v es un *nodo hoja*.

Definición 14 *Dado $\mathcal{H}$, un hiperarco $e \in \mathcal{E}$ es una tupla $(H(e), T(e), t(e), p(e))$ donde la cola $T(e)$ y la cabeza $H(e)$ son subconjuntos de $\mathcal{V}$, $t(e)$ es una transcripción asociada con el hiperarco y $p(e)$ es una puntaje (o peso).*

Dado que las GIP-2D consideradas están en *Forma Normal de Chomsky*, la cabeza $H(e)$ contiene exactamente un nodo (interno) y la cola $T(e)$ dos nodos (internos, para modelar reglas binarias, $A \rightarrow BC$) o un nodo (hoja, para modelar reglas unarias, $A \rightarrow z_i$). Éste es un caso particular de hiperarco *B-arc* definido en in [8]. $t(e)$ es la transcripción (LaTeX, en nuestro caso) representado la semántica asociada con el arco Finalmente, $p(e)$ es el puntaje del arco. En nuestro caso, el puntaje del arco representa una probabilidad. Dependiendo en si e representa una regla binaria o unaria, su probabilidad se obtiene de Eq. (2.2) o Eq. (2.3) respectivamente.

Definición 15 *Un árbol completo $\boldsymbol{t}$ de un hipergrafo $\mathcal{H}$ es una secuencia de hiperarcos en donde hay un nodo raíz (y solo uno) que cubre completamente la EM de entrada representada por $\boldsymbol{z}$.*

Debe haber un hiperarco, y solo uno, e en t que cumple: $n(H(e)) = S \in \mathcal{N}$, donde S es el axioma de la GIP-2D; y $s(H(e)) = \boldsymbol{z}$, donde el span de $\boldsymbol{z}$ es el conjunto completo de componentes conectadas que representa la EM de entrada. La probabilidad de un árbol es el producto de las probabilidades de todos los hiperarcos que lo componen. Por lo tanto, dado un conjunto de componentes conectadas $\boldsymbol{z}$, la probabilidad conjunta $p(\boldsymbol{t}, \boldsymbol{z})$ puede apro-

ximarse de un hipergrafo $\mathcal{H}$ como:

$$p(\boldsymbol{t}, \boldsymbol{z}) \approx \prod_{e \in \psi(\boldsymbol{t})} p(e) \stackrel{\text{def}}{=} p_{\mathcal{H}}(\boldsymbol{t}, \boldsymbol{z}) \tag{3.4}$$

donde $\psi(\boldsymbol{t})$ denota el conjunto de hiperarcos del árbol completo $\boldsymbol{t}$ en $\mathcal{H}$. Un hipergrafo $\mathcal{H}$ típicamente debería contener la mayoría de las hipótesis de decodificación más probables consideradas en la maximización Eq. (3.3), incluida la mejor hipótesis. Por lo tanto, la probabilidad incondicional de $\boldsymbol{z}$ puede aproximarse con la probabilidad acumulativa de todos los árboles completos representados en $\mathcal{H}$:

$$p(\boldsymbol{z}) \approx \sum_{\boldsymbol{t}} p_{\mathcal{H}}(\boldsymbol{t}, \boldsymbol{z}) \stackrel{\text{def}}{=} p_{\mathcal{H}}(\boldsymbol{z}) \tag{3.5}$$

Ésta expresión puede ser calculada eficientemente usando programación dinámica, como se ve en la Sec. 3.3.

3.2.2. Generación de hipergrafos

Una vez los n-mejores árboles de análisis para una expresión de entrada son generados, para obtener el hipergrafo completo $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ necesitamos recorrer cada árbol, guardando todos los nodos y reglas en una estructura de grafo, combinando terminales y pre-terminales que representan el mismo conjunto de componentes conectadas. Desde los N-mejores árboles de análisis para una EM de entrada, representados por $\boldsymbol{z}$, el algoritmo en la figura 3.6 calcula el conjunto de nodos e hiperarcos del hipergrafo $\mathcal{H}$

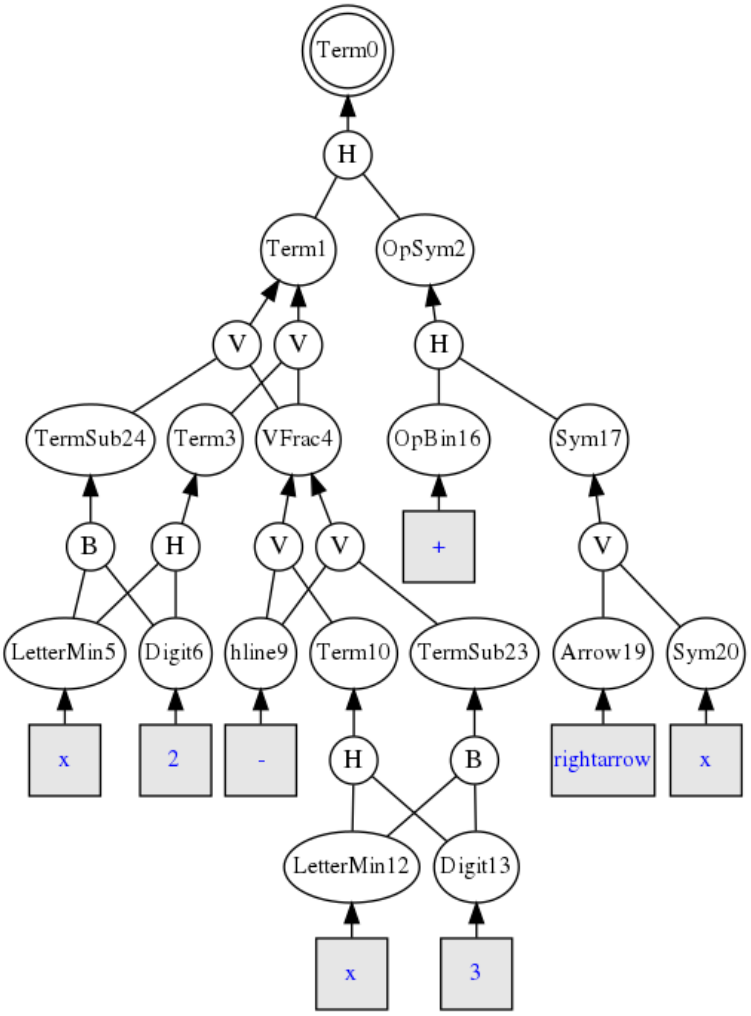


Figura 3.5: Hipergrafo para "$\frac{x_2}{x_3} + \vec{x}$" con 3-mejores soluciones.

La figura 3.5 muestra el hipergrafo obtenido al aplicar el algoritmo 3.6 en los 3 mejores árboles completos de la expresión "$\frac{x_2}{x_3} + \vec{x}$" que se ven en 3.2.

```
Calcular:  T^1(S,z),...,T^n(S,z)
V={}   # Nodos
Hu={}  # Hiperarcos unarios
Hb={}  # Hiperarcos binarios

for all T^i(S,z) in T^1(S,z),...,T^N(S,z):
  addTree(T^i(S,z))
return (V,[Hu+Hb])

def addTree(⟨T(A,a),T^p(B,b),T^q(C,c)⟩):
  i = addNode(A,a)
  if |a| == 1:
    j = addNode(s,a)
    if (i(A) → j("s"),p) not in Hu:
      Hu[|Hu|] = (i(A) → j("s"),p))
  else:
    j = addNode(B,b)
    k = addNode(C,c)
    if (i(A) → j(B)k(C),p)) not in Hb:
      Hb[|Hb|] = (i(A) → j(B)k(C),p))
    addTree(T^p(B,b))
    addTree(T^q(C,c))

def addNode(X,x):
  for i = 0...|V|:
    if (X,x) in V[i]:
      return i
  V[|V|] = (X,x)
  return |V|-1
```

Figura 3.6: Algoritmo para generar un hipergrafo usando los n-mejores árboles de análisis

3.3. Algoritmos de inferencia en hipergrafos

Una forma eficiente de calcular la probabilidad total de Eq. (3.5) es generalizar los conocidos algoritmos de Inside y Outside para GIPs [18] y extender los para GIP-2D e hipergrafos.

3.3.1. Algoritmo *Inside* para hipergrafos

Dado $\mathcal{H}$ y el conjunto de componentes conectadas $\boldsymbol{z}$, asociada con la EM de entrada, por cada uno de los nodos $v \in \mathcal{V}$ de $\mathcal{H}$, podemos definir

$$\alpha_{\mathcal{H}}(A, a) \stackrel{\text{def}}{=} p(A \stackrel{*}{\Rightarrow} a), \tag{3.6}$$

como la probabilidad de que $A = n(v)$ es una solución al conjunto de componentes conectadas, $a = s(v)$.

Initialización. De acuerdo a Eq. (2.2) y considerando adición en vez de maximización. $\forall v = (A, a) \in \mathcal{V}$, with $a = \{z_i\}$, and $1 \leq i \leq |\boldsymbol{z}|$,

$$\alpha_{\mathcal{H}}(A, \{z_i\}) \;=\; \frac{1}{|\boldsymbol{z}|} \sum_{s} \frac{p(A \to s)\; p(s \mid \{z_i\})}{p(s)}$$

Recursión. De acuerdo a Eq. (2.3), y considerando adición en vez de maximización. $\forall v = (A, a) \in \mathcal{V}$,

$$\alpha_{\mathcal{H}}(A, a) = \sum_{\substack{e \in \mathcal{E} \\ r}} \alpha_{\mathcal{H}}(B, b)\; \alpha_{\mathcal{H}}(C, c)\; p(A \to BC)\; p(r|BC)$$

donde $e \in \mathcal{E}$ satisface que $H(e) = v$ y $T(e) = (m, n)$, con $v, m, n \in \mathcal{V}$, donde $n(v) = A, n(m) = B$ and $n(n) = C$ son los tags de nodo y $s(v) = a, s(m) = b,$ and $s(n) = c$ son los spans de nodo, que tiene que satisfacer $b \cap c = \emptyset$ y $b \cup c = a$. Finalmente, la probabilidad $p_{\mathcal{H}}(z)$, definida en la Eq. (3.5), puede calcularse como,

$$p_{\mathcal{H}}(z) \;=\; \alpha_{\mathcal{H}}(S, \boldsymbol{z}) \tag{3.7}$$

donde S es el axioma de la gramática y $\boldsymbol{z}$ es el conjunto de componentes conectadas de la EM de entrada.

3.3.2. Algoritmo *Outside* para hipergrafos

Similarmente, dado $\mathcal{H}$ y el conjunto de componentes conectadas $\boldsymbol{z}$, asociadas con la EM de entrada, por cada uno de los nodos $v \in \mathcal{V}$ de $\mathcal{H}$, podemos definir

$$\beta_{\mathcal{H}}(A, a) \stackrel{\text{def}}{=} p(S \stackrel{*}{\Rightarrow} g\, A\, u), \tag{3.8}$$

como la probabilidad de que $A = n(v)$ sea una solución del conjunto de componentes conectadas, $a = s(v)$, y el resto de $\mathcal{H}$ explica correctamente las componentes conexas que no están en a, donde g y u son dos conjuntos disjuntos de componentes conectadas que cumplen que $g \cup a \cup u = \boldsymbol{z}$.

Initialización. Para el (único) nodo raíz de $\mathcal{H}$, donde el tag es el axioma S de la gramática, y $\boldsymbol{z}$ es el conjunto completo de componentes conectadas asociadas con el EM de entrada, definimos

$$\beta_{\mathcal{H}}(S, \boldsymbol{z}) \;=\; 1$$

Recursión. $\forall v = (A, a) \in \mathcal{V}$,

$$\begin{aligned}
&\beta_{\mathcal{H}}(A, a) \;= \\
&\quad \sum_{\substack{e \in \mathcal{E} \\ r}} \beta_{\mathcal{H}}(B, b)\; \alpha_{\mathcal{H}}(C, c)\; p(B \to AC)\; p(r|AC) \;+ \\
&\quad \sum_{\substack{e' \in \mathcal{E} \\ r}} \beta_{\mathcal{H}}(B, b)\; \alpha_{\mathcal{H}}(C, c)\; p(B \to CA)\; p(r|CA)
\end{aligned}$$

donde $e, e' \in \mathcal{E}$ satisfaces respectivamente que $H(e) = H(e') = m$, $T(e) = (v, n)$ y $T(e') = (n, v)$ con $v, m, n \in \mathcal{V}$. Además, $n(v) = A, n(m) = B,$ and $n(n) =$

C son los tags de nodo y $s(v) = a, s(m) = b,$ and $s(n) = c$ son los span de nodos. Finalmente, la probabilidad $p_{\mathcal{H}}(z)$, definida en Eq. 3.5, puede calcularse también como,

$$p_{\mathcal{H}}(\boldsymbol{z}) \;=\; \sum_{e \in \mathcal{E}} \beta_{\mathcal{H}}(A, \{z_i\}) \; \frac{1}{|\boldsymbol{z}|} \; \sum_{s} \; \frac{p(A \rightarrow s) \; p(s \mid \{z_i\})}{p(s)}$$

Para $1 \leq i \leq |\boldsymbol{z}|$, donde $H(e) = v = (A, \{z_i\})$.

3.4. Normalización de hipergrafos

En esta sección vamos a abordar el problema de la normalización de los hiperarcos de un hipergrafo. Para hacerlo, empezamos definiendo la probabilidad posterior de un árbol asociado con un hipergrafo $\mathcal{H}$.

Definición 16 *Dado un hipergrafo $\mathcal{H}$, la probabilidad posterior del árbol $p(\boldsymbol{t}|\boldsymbol{z})$ puede calcularse aproximadamente como:*

$$p(\boldsymbol{t}|\boldsymbol{z}) \;\approx\; \frac{p_{\mathcal{H}}(\boldsymbol{t}, \boldsymbol{z})}{p_{\mathcal{H}}(\boldsymbol{z})} \;\stackrel{def}{=}\; p_{\mathcal{H}}(\boldsymbol{t}|\boldsymbol{z}) \tag{3.9}$$

En una forma similar ha como se introduce en [29] en el caso de lattices de palabras, los puntajes (o probabilidades) de hiperarcos de un hipergrafo pueden normalizarse de distintas maneras. Para este trabajo, necesitamos los puntajes de hiperarco normalizados de la siguiente forma:

Definición 17 *Para un hiperarco dado $e \in \mathcal{E}$, su puntaje normalizado, $\varphi(e)$, se define como la suma de probabilidad posteriores de todos los árboles de análisis completos que incluyen el arco e. Esto es:*

$$\varphi(e) \;\stackrel{def}{=}\; \sum_{\boldsymbol{t}: e \in \psi(\boldsymbol{t})} p_{\mathcal{H}}(\boldsymbol{t}|\boldsymbol{z}) \tag{3.10}$$

Un ejemplo de hipergrafos normalizados puede verse en la figura 3.5. Para un hipergrafo normalizado de esta manera, pueden demostrarse las siguientes propiedades:

Teorema 1 *Dado un hipergrafo $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, los puntajes normalizados de un hiperarco $\varphi(e) : e \in \mathcal{E}$ puede ser calculado eficientemente por la expresión,*

$$\varphi(e) \;=\; \frac{\alpha_{\mathcal{H}}(A,a)\ \beta_{\mathcal{H}}(A,a)}{\alpha_{\mathcal{H}}(S,z)} \tag{3.11}$$

Definición 18 *Dado $\mathcal{H}$, $\forall e \in \mathcal{E}$, Eq.* (3.10) *indica que la suma debe hacerse para todos los árboles de análisis $\boldsymbol{t}$ in $\mathcal{H}$, donde el hiperarco e aparece en una posición concreta. Por lo tanto, considerando Eq.* (3.9) *y Eq.* (3.7)*, entonces, Eq.* (3.10) *puede escribirse como:*

$$\sum_{\boldsymbol{t}:e\in\psi(\boldsymbol{t})} p_{\mathcal{H}}(\boldsymbol{t}|\boldsymbol{z}) \;=\; \frac{1}{\alpha_{\mathcal{H}}(S,\boldsymbol{z})} \sum_{\boldsymbol{t}:e\in\psi(\boldsymbol{t})} p_{\mathcal{H}}(\boldsymbol{t},\boldsymbol{z}) \tag{3.12}$$

donde $p_{\mathcal{H}}(\boldsymbol{t},\boldsymbol{z})$ es el producto de probabilidades de los hiperarcos del árbol de análisis $\boldsymbol{t}$, con la restricción de que el hiperarco e esta presente en $\boldsymbol{t}$. Dado $H(e) = v$ es el nodo cabeza, donde $n(v) = A$ es el tag de nodo y $s(v) = a$ es el span asociado con el nodo, tenemos que:

$$\sum_{\boldsymbol{t}:e\in\psi(\boldsymbol{t})} p_{\mathcal{H}}(\boldsymbol{t},\boldsymbol{z}) \;=\; p(S \overset{*}{\Rightarrow} g\,A\,u)\ p(A \overset{*}{\Rightarrow} a)$$

donde $g \cap a \cap u = \emptyset$ y $g \cup a \cup u = \boldsymbol{z}$.

Considerando las definiciones de probabilidad inside (Eq. (3.6)) y outside (Eq. (3.8)) y el puntaje normalizado del arco (Eq. 3.10), la Eq. (3.12) puede reescribirse como,

$$\sum_{\boldsymbol{t}:e\in\Upsilon(\boldsymbol{t})} p_{\mathcal{H}}(\boldsymbol{t}|\boldsymbol{z}) \;=\; \frac{\alpha_{\mathcal{H}}(A,a)\ \cdot\ \beta_{\mathcal{H}}(A,a)}{\alpha_{\mathcal{H}}(S,z)} \;=\; \varphi(e)$$

Capítulo 4

Evaluación

Una vez presentados todos los principios y algoritmos requeridos en el primer capítulo, en este capítulo veremos la implementación de un programa en C++ capaz de obtener el hipergrafo de una imagen de una expresión matemática, y utilizaremos esta implementación para evaluar varias métricas que nos permitirán juzgar su utilidad en la indexación de documentos.

4.1. Seshat C++

El programa desarrollado acepta como entrada una imagen de una expresión matemática. Primero segmenta las componentes conexas y utiliza una red neuronal convolucional para clasificarlas en uno de los símbolos reconocidos, segundo utiliza una GIP-2D personalizable y un modelo de mixturas de gaussianas junto con el algoritmo CYK-2D visto en la Figura 2.3 para obtener la mejor derivación y su conjunto de símbolos; teniendo la 1-mejor derivación e implementando estructuras de datos para guardar y obtener eficientemente los sub-árboles visitados durante la etapa previa, usamos el algoritmo de la figura 3.3 para obtener los siguientes (N-1) mejores árboles de análisis y al final el algoritmo de la figura 3.6 para combinar

los N árboles en la estructura de hipergrafo vista en la sección anterior.

4.1.1. Evaluación del modelo de reconocimiento de símbolos matemáticos

Para la implementación de la Red Neuronal Convolucional(CNN), hemos utilizado la librería DLIB de C++ para crear una red con la siguiente arquitectura:

Capa	Tipo	Parámetros
layer<0>	multiclass-log	
layer<1>	fc	(outputs=165)
layer<2>	relu	
layer<3>	fc	(outputs=84)
layer<4>	relu	
layer<5>	fc	(outputs=120)
layer<6>	max-pool	(nr=nc=2, stride-y=stride-x=2, padding-y=padding-x=0)
layer<7>	relu	
layer<8>	con	(filters=16, nr=nc=5, stride-y=stride-x=1, padding-y=padding-x=2)
layer<9>	max-pool	(nr=nc=2, stride-y=stride-x=2, padding-y=padding-x=0)
layer<10>	relu	
layer<11>	con	(filters=6, nr=nc=5, stride-y=stride-x=1, padding-y=2padding-x=2)
layer<12>	input<matrix>	

Cuadro 4.1: Arquitectura del modelo de Red Neuronal Convolucional

Para entrenar el modelo, fue necesario crear un dataset de imágenes de las distintas componentes conexas, este dataset puede generarse implementando un programa que imprime un símbolo de LaTeXen una imagen y aplica una leve transformación. Una vez implementado el programa hemos generado 1.000 imágenes por clase, cada una con distintas transformaciones de zoom, rotación y dilatación aplicadas. De las 165.000 imágenes generadas en total, separamos 120.000 como muestras de entrenamiento para el modelo y las 45.000 restantes son utilizadas para evaluar el modelo. En la Figura 4.1 podemos ver las 165 clases reconocidas por el modelo.

Figura 4.1: Símbolos aceptados por el reconocedor.

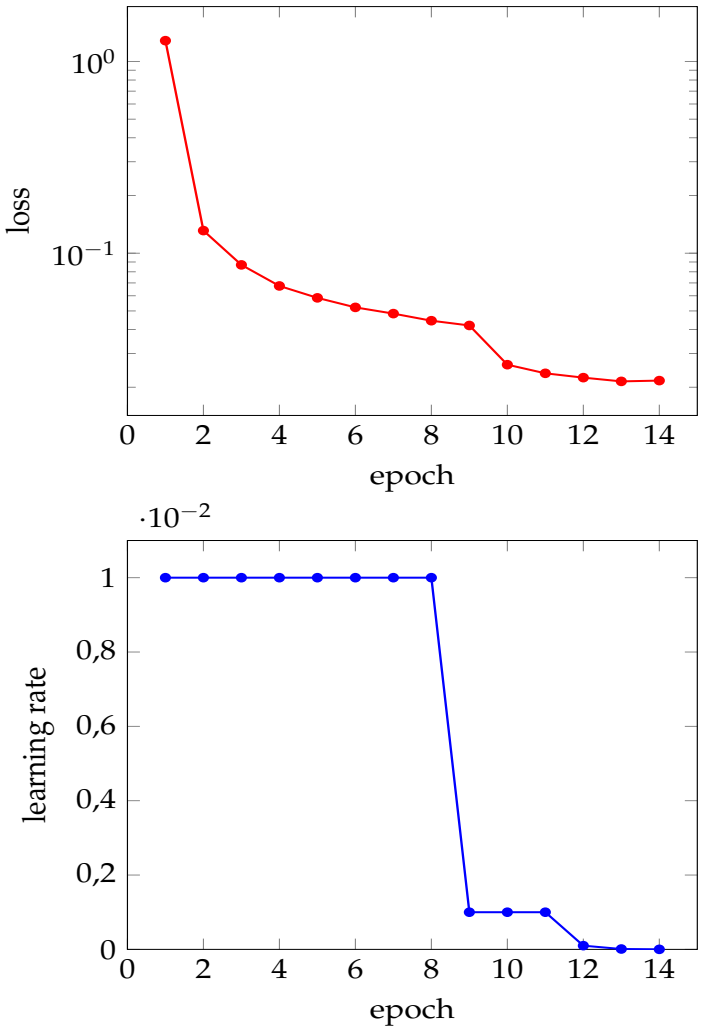


Figura 4.2: Evolución de la función de pérdida y el factor de aprendizaje durante el entrenamiento del modelo

	Total	Nº Aciertos	Nº Errores	Precisión
training	120.000	118824	1176	0,9902
testing	45.000	44304	696	0,9845

Cuadro 4.2: Resultado del entrenamiento del modelo de reconocimiento de símbolos

4.1.2. Evaluación de la precisión dentro de las N-mejores derivaciones

Como hemos visto en la sección 1.2.2, la mayoría de enfoques propuestos para resolver el problema de reconocimiento automático de expresiones matemáticas no pueden compararse debidamente por la falta de datasets públicos y métricas estandarizadas.

Para obtener una evaluación de la precisión del programa implementado, en esta sección utilizaremos un dataset de 1590 expresiones matemáticas codificadas en LaTeXque ha sido recolectado dentro del marco del proyecto IBEM. Cada expresión es renderizada a una imagen, proporcionándonos 1590 imágenes de EM cada una con una de sus posibles derivaciones.

Dado que, como hemos mostrado anteriormente, es posible escribir la misma EM con distintas codificaciones de LaTeXe incluso la misma codificación puede tener distintos caracteres especiales usados, una evaluación directa de la expresión de entrada vs la generada por el programa podría descartar soluciones propuestas que a pesar de no codificarse de igual manera en LaTeX, describen la imagen de entrada correctamente.

Por ésto, en este trabajo veremos dos medidas distintas para evaluar el rendimiento del trabajo, una basada en la comparaciones de las codificaciones LaTeXde las entradas y otras en la comparación de las imágenes renderizadas de dichas expresiones. Además, para mostrar el efecto de utilizar varias derivaciones además de la mejor en la generación del resultado, realizamos distintas pruebas de rendimiento para varios valores de N.

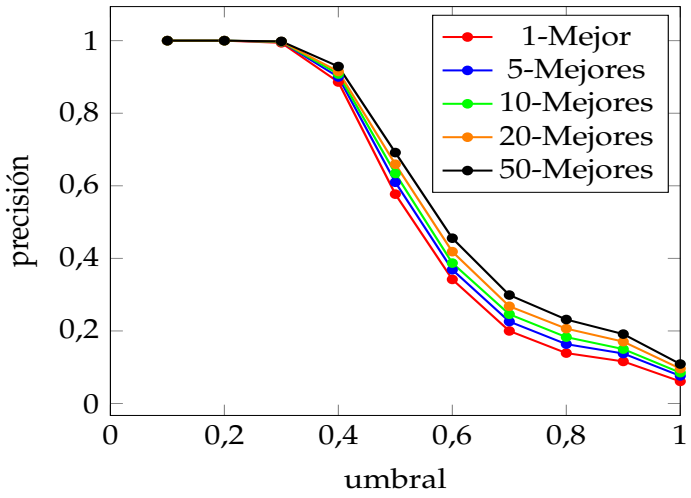


Figura 4.3: Evolución de la precisión para distintos valores del umbral de distancia SSIM entre imágenes

	Umbral	Nº Aciertos	Nº Errores	Precisión
1-Mejor	0,75	266	1324	0,1672
5-Mejores	0,75	306	1284	0,1924
10-Mejores	0,75	335	1255	0,2106
20-Mejores	0,75	371	1219	0,2333
50-Mejores	0,75	420	1170	0,2641

La figura 4.3 muestra la evolución de la precisión del programa si consideramos correctas las imágenes cuyo Indice de similitud estructural(SSMI) [32, 31] sea mayor a un umbral dado.

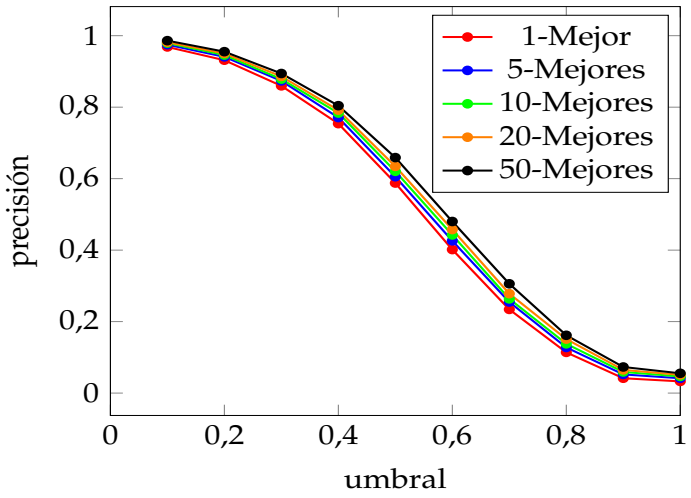


Figura 4.4: Evolución de la precisión para distintos valores del umbral de distancia bleu entre expresiones LaTeX

	Umbral	Nº Aciertos	Nº Errores	Precisión
1-Mejor	0,75	263	1327	0,1654
5-Mejores	0,75	299	1291	0,1880
10-Mejores	0,75	319	1271	0,2006
20-Mejores	0,75	337	1253	0,2119
50-Mejores	0,75	368	1222	0,2314

La figura 4.4 muestra la evolución de la precisión del programa si consideramos correctas las expresiones LaTeXcuya distancia BLEU [22] sea mayor a un umbral dado.

4.1.3. Evaluación de la complejidad temporal de los algoritmos introducidos

En esta sección estudiamos la complejidad temporal de los algoritmos dependiendo del tamaño de la EM y el número de árboles generados para

crear el hipergrafo. Para evaluar el efecto del tamaño de la expresión y el número de relaciones entre componentes conectadas en el tiempo de ejecución, usamos dos tipos distintos de EM para ilustrar los mejores y peores escenarios de estas variables. Las figuras 4.5 y 4.6 fueron creadas usando expresiones simples, donde solo consideramos relaciones espaciales horizontales del tipo $x + ... + x_s$ La figura 4.7 fue realizada usando un esquema más complejo donde la EM crece tanto horizontalmente como verticalmente utilizando combinaciones de sumas y fracciones, dando un caso más pesimista para el tiempo de ejecución. Al incrementar el número de árboles de análisis generados para una EM, podemos estudiar el efecto en una expresión pequeña con reducido número de árboles de análisis distintos, y expresiones largar con más árboles posibles. [1]

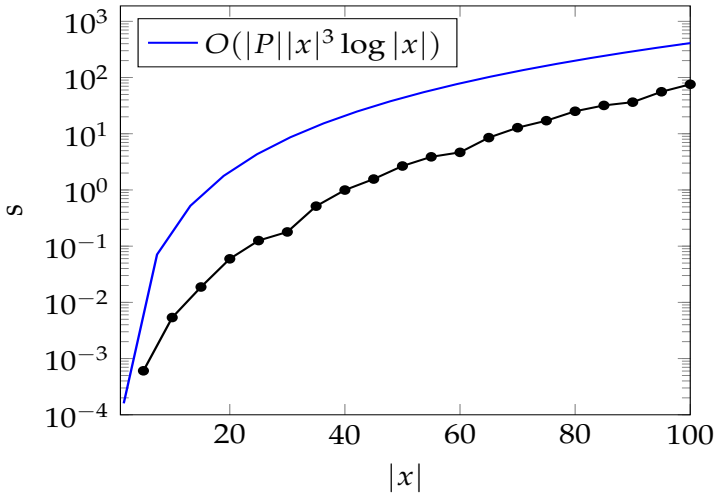


Figura 4.5: Mejor derivación para $|x|$=[5, 100] en relaciones horizontales.

La figura 4.5 muestra el tiempo de ejecución medio al calcular el mejor árbol de análisis en EM de longitud incremental. El eje de ordenadas representan el tiempo de ejecución en segundos en una escala logarítmica y

[1]Todos los experimentos fueron realizados en una computadora Intel(R) Core(TM) i5-6600K CPU de 3 GHz corriendo Ubuntu 18.04. El algoritmo ha sido implementado en C++ y compilado con g++ 7.5.0. -O3

el eje de abscisas representa el tamaño $|z|$ de la EM de entrada. El gráfico muestra en azul la complejidad temporal teórica y el comportamiento real se muestra con una linea negra punteada.

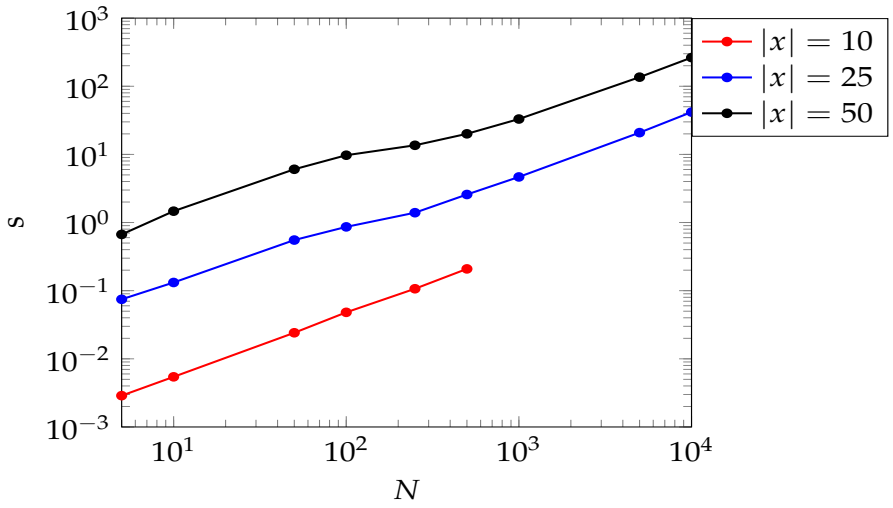


Figura 4.6: Tiempo de ejecución para calcular N-mejor para distintos valores de N en expresiones de distintos tamaños de $|x|$. La expresión roja de $|x| = 10$ tiene un máximo de 500 hipótesis

La figura 4.6 muestra el efecto de incrementar el número de árboles de análisis a generar, N, en EM simples de tamaños $10, 25$ y 50. El eje de ordenadas representa el tiempo de ejecución en segundos y el eje de abscisas representa el número de árboles de análisis que son generados, ambos en escala logarítmica. El tiempo en calcular la 1-mejor solución es excluido de este gráfico.

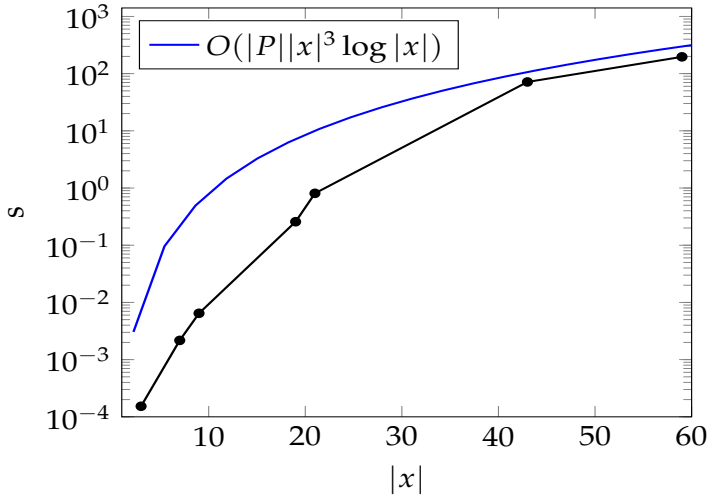


Figura 4.7: Mejor derivación para $|x| = \{3, 7, 9, 19, 21, 43, 59\}$ en relaciones horizontales y verticales.

La figura 4.7 muestra el tiempo de ejecución al calcular el 1-mejor árbol para EM de tamaño incremental complejas. El eje de ordenadas representa el tiempo de ejecución en segundo en escala logarítmica y el eje de abscisas representa el tamaño $|x|$ de la EM de entrada. Como en el experimento previo, la gráfica muestra en azul la complejidad temporal teórica y el comportamiento real se muestra con una linea negra punteada. Estos experimentos muestran que el algoritmo de análisis para 1-mejor es el procedimiento más caro.

4.2. Parametrización de GIP-2D

Gracias a la implementación del programa anterior que nos permite obtener las N-mejores derivaciones de una EM en formato de árbol sintáctico, en esta sección utilizaremos el planteamiento presentado en 3.1.2 para implementar un proceso que utilizando el dataset de expresiones en LaTeXpresentado anteriormente, calcula iterativamente mejores pesos para

la GIP-2D utilizada. Para evaluar el efecto de ajustar los pesos de la gramática usando un conjunto de expresiones, hemos dividido las 1590 expresiones LaTeXdel dataset en un conjunto de entrenamiento de 1190 EM que se utiliza para ajustar los pesos de la gramática y otro conjunto de 400 expresiones de validación que no se ven durante el cálculo de los pesos.

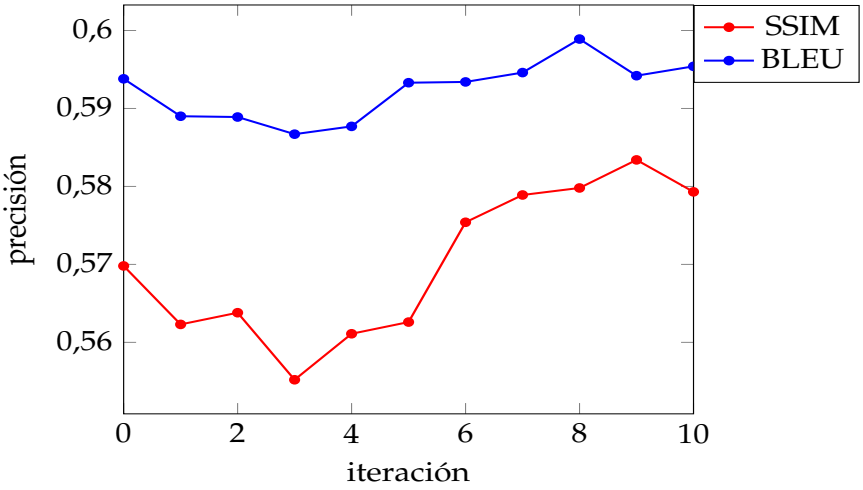


Figura 4.8: Evolución de la distancia media entre la EM de entrada y la media de las 10-Mejores soluciones en el conjunto de entrenamiento, usando SSIM para comparar las imágenes generadas y BLEU para las expresiones LaTeX

En la figura 4.8 se ve que la distancia media dentro de las 10-Mejores soluciones de EM del conjunto de entrenamiento mejora poco al final de las 10 iteraciones realizadas del mismo. La mejora poco significativa puede deberse a un factor de aprendizaje pequeño, o similarmente a que se requieren muchas más iteraciones para converger en pesos ajustados. También podemos ver que los cambios producidos afectan más la comparación de imágenes que la de expresiones textuales.

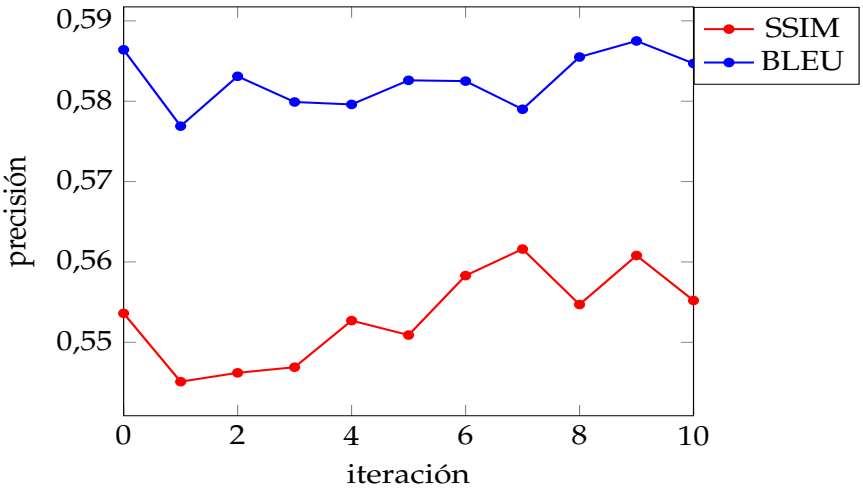


Figura 4.9: Evolución de la distancia media entre la EM de entrada y la media de las 10-Mejores soluciones en el conjunto de validación, usando SSIM para comparar las imágenes generadas y BLEU para las expresiones LaTeX

En la figura 4.9 vemos que la distancia media dentro de las 10-Mejores soluciones de EM del conjunto de validación no tiene una mejora concluyente durante las 10 iteraciones del conjunto de entrenamiento.

Capítulo 5

Conclusiones

5.1. Conclusión

En este trabajo hemos estudiado el problema de reconocimiento automático de expresiones matemáticas en documentos impresos, hemos visto definiciones usadas en el campo y distintas estrategias propuestas con sus ventajas y deficiencias, hemos extendido algunas de éstas estrategias, resolviendo las distintas dificultades encontradas y definiendo modelos útiles tanto para la segmentación y clasificación de símbolos como para él análisis estructural de las expresiones, finalmente hemos presentado una propuesta propia para generar hipergrafos desde el análisis de N-mejores en GIP-2D.

Más específicamente, los principales logros obtenidos en este trabajo son:

1. Adaptar un algoritmo para calcular el mejor árbol de análisis desde una GIP-2D para imágenes de documentos científicos impresos.

2. Desarrollar un algoritmo para calcular los N-mejores árboles de análisis desde una GIP-2D.

3. Definir e implementar un algoritmo para representar los N-mejores árboles de análisis de forma compacta usando hipergrafos.

4. Proponer un marco formal para el desarrollo de algoritmos de inferencia (INSIDE y OUTSIDE) y estrategias de normalización de hipergrafos.

Adicionalmente, desarrollamos un prototipo que implementa los algoritmos descritos y ejecutamos algunos experimentos preliminares para comprobar el comportamiento de los mismos.

Las evaluaciones realizadas nos indican que los hipergrafos pueden ser interfaz eficiente entre reconocimiento de expresiones matemáticas y búsqueda e indexación de sistemas de expresiones matemáticas. Los resultados de generar las N-mejores soluciones necesarias para crear los hipergrafos escalan de forma lineal y en EM realistas de longitud $|x| = 25$ el tiempo necesario para calcular los primeros 100 árboles se encuentra cerca de 1 segundo, siendo posible todavía optimizar aspectos ne la implementación de los algoritmos, sería posible su uso en aplicaciones reales.

5.2. Trabajos futuros

Siguiendo el objetivo del trabajo, hemos definido un marco estadístico que permite en el futuro utilizar el enfoque presentado por HIMANIS [29] para generar índices y representaciones intermedias que puedan usarse en aplicaciones reales para la búsqueda e indexación de expresiones matemáticas en grandes colecciones de imágenes digitalizadas.

Otra tarea que no se ha podido terminar en el alcance de este trabajo consiste en optimizar los hiperparámetros de los modelos y la implementación de los algoritmos usados dentro del código C++, buscando mejorar la precisión global del sistema y reducir el tiempo de ejecución. Dentro de esta tarea, y dadas nuestras experiencias durante este trabajo, creemos recomendable enfocarse en mejorar la gramática GIP-2D utilizada, podando reglas innecesarias y calculando pesos óptimos para las restantes, y además mejorar el modelo de mixturas gaussianas utilizado para obtener las probabilidades de las distintas relaciones espaciales.

5.3. Contribuciones

Éste trabajo se ha desarrollado dentro del proyecto IBEM:

"Indexación y búsqueda de expresiones matemáticas a gran escala en corpus masivos de documentos impresos". Financiado por el Ministerio de Ciencia, Innovación y Universidades (TIN2017-91452-EXP). 1-11-2018 a 31-12-2020.

5.4. Publicaciones

El contenido de este trabajo lo hemos presentado al ICPR 2021. Ha pasado la primera ronda de un sistema de revisión de dos rondas y actualmente se encuentra en proceso de revisión:

- Ernesto Noya, Joan Andreu Sánchez y José Miguel Benedí. «Generation of Hypergraphs from the N-Best Parsing of 2D-Probabilistic Context-Free Grammars for Mathematical Expression Recognition». En: *25th International Conference on Pattern Recognition (ICPR2020)* (2021)

Agradecimientos

Este trabajo ha sido parcialmente financiado por el Ministerio de Ciencia y Tecnología en el proyecto IBEM (TIN2017-91452-EXP) y por la Generalitat Valenciana en el proyecto DeepPattern (PROMETEO/2019/121).

Índice de figuras

Bibliografía

[1] R.H. Anderson. «Syntax-directed recognition of hand-printed two-dimensional mathematics». En: *ACM Symposium on Interactive Systems for Experimental Applied Mathematics* (1967), págs. 436-459.

[2] A.M. Awal, H. Mouchére y C. Viard-Gaudin. «A global learning approach for an online handwritten mathematical expression recognition system». En: *Pattern Recognit. Lett. 35* (2014), págs. 68-77.

[3] K. Chan y D. Yeung. «Mathematical expression recognition: a survey». En: *Int. J. Doc. Anal. Recognit.* (2000), págs. 3-15.

[4] K.F. Chan y D.Y. Yeung. «Error detection, error correction and performance evaluation in on-line mathematical expression recognition». En: *Pattern Recognition* (2001), págs. 1671-1684.

[5] P.A. Chou. «Recognition of equations using a two-dimensional stochastic context-free grammar». En: *Vis. Commun. Image Process. IV 1199* (1989), págs. 852-863.

[6] D.Klein y C.D.Manning. «Parsing and hypergraphs». En: *Proc. IWPT* (2001).

[7] Y. Eto y M. Suzuki. «Mathematical formula recognition using virtual link network». En: *International Conference on Document Analysis and Recognition* (2001), págs. 762-767.

[8] Giorgio Gallo, Giustino Longo, Stefano Pallottino y Sang Nguyen. «Directed Hypergraphs and Applications». En: *Discrete Appl. Math.* 42.2–3 (1993), págs. 177-201.

[9] L. Hu y R. Zanibbi. «HMM-based recognition of online handwritten mathematical symbols using segmental K-means initialization and a modified pen-up/down feature». En: *International Conference on Document Analysis and Recognition* (2011), págs. 457-462.

[10] V. Jimenez y A. Marzal. «Computation of the n-best parse trees for weighted and stochastic context-free grammars». En: *LNCS-1876. Springer-Verlag* (2000), págs. 183-192.

[11] Víctor Jiménez y Andrés Marzal. «Computation of the N Best Parse Trees for Weighted and Stochastic Context-Free Grammars». En: *Advances in Pattern Recognition*. Ed. por Francesc Ferri, José Iñesta, Adnan Amin y Pavel Pudil. Springer Berlin Heidelberg, 2000, págs. 183-192.

[12] B. Keshari y S. Watt. «Hybrid mathematical symbol recognition using support vector machines». En: *International Conference on Document Analysis and Recognition* (2007), págs. 859-863.

[13] S. Lavirotte y L. Pottier. «Mathematical formula recognition using graph grammar». En: *Proceedings of the SPIE* (1998), págs. 44-52.

[14] Y. LeCun, B. Boser, y J. S. Henderson. «Handwritten digit recognition with a back-propagation network». En: *Advances in Neural Information Processing Systems* (1990), págs. 396-404.

[15] Z. Luo, Y. Shi y F.K. Soong. «Symbol graph based discriminative training and rescoring for improved math symbol recognition». En: *IEEE International Conference on Acoustics, Speech, and Signal Processing* (2008), págs. 1953-1956.

[16] S. MacLean y G. Labahn. «A new approach for recognizing handwritten mathematics using relational grammars and fuzzy sets». En: *Int. J. Doc. Anal. Recognit. 16* (2013), págs. 139-163.

[17] S. MacLean y G. Labahn. «Elastic matching in linear time and constant space». En: *IAPR International Workshop on Document Analysis Systems* (2010), págs. 551-554.

[18] Christopher D. Manning e Hinrich Schutze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[19] U.V. Marti y H. Bunke. «Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system». En: *Int. J. Pattern Recognit. Artif. Intell.* (2001), págs. 65-90.

[20] Ernesto Noya, Joan Andreu Sánchez y José Miguel Benedí. «Generation of Hypergraphs from the N-Best Parsing of 2D-Probabilistic Context-Free Grammars for Mathematical Expression Recognition». En: *25th International Conference on Pattern Recognition (ICPR2020)* (2021).

[21] Stefan Ortmanns, Hermann Ney y Xavier Aubert. «A word graph algorithm for large vocabulary continuous speech recognition». En: *Computer Speech and Language* 11.1 (1997), págs. 43-72.

[22] Kishore Papineni, Salim Roukos, Todd Ward y Wei jing Zhu. «BLEU: a Method for Automatic Evaluation of Machine Translation». En: *40th Annual meeting of the Association for Computational Linguistics* (2002), págs. 311-318.

[23] B. Pearlmutter. «Learning state space trajectories in recurrent neural networks». En: *International Joint Conference on Neural Networks* (1989), págs. 365-372.

[24] R. Plamondon y S. Srihari. «Online and off-line handwriting recognition: a comprehensive survey». En: *IEEE Trans. Pattern Anal. Mach. Intell.* (2000), págs. 63-84.

[25] Y. Shi, H. Li y F.K. Soong. «A unified framework for symbol segmentation and recognition of handwritten mathematical expressions». En: *International Conference on Document Analysis and Recognition* (2007), págs. 854-858.

[26] Y. Shi y F. Soong. «A symbol graph based handwritten math expression recognition». En: *International Conference on Pattern Recognition* (2008), págs. 1-4.

[27] A. Thammano y S. Rugkunchon. «A neural network model for online handwritten mathematical symbol recognition». En: *Intell. Comput.* (2006), págs. 292-298.

[28] A.H. Toselli. «Integrated handwriting recognition and interpretation using finite-state models». En: *Int. J. Pattern Recognit. Artif. Intell.* (2004), págs. 519-539.

[29] A.H. Toselli, E. Vidal, V. Romero y V. Frinken. «HMM word graph based keyword spotting in handwritten document images». En: *Information Sciences* (2016), 497-518.

[30] E. Vidal. «Text Search and Information Retrieval in Large Historical Collections of Untranscribed Manuscripts». En: *Invited key note talk at International Conference on Document Analysis and Recognition (ICDAR)* (2019).

[31] Z. Wang, A.C. Bovik, H.R. Sheikh y E.P. Simoncelli. «Image quality assessment: From error visibility to structural similarity». En: *IEEE Transactions on Image Processing* (2004), págs. 600-612.

[32] Zhou Wang y A.C. Bovik. «Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures». En: *Signal Processing Magazine, IEEE* (2009), págs. 98-117.

[33] H.J. Winkler. «HMM-based handwritten symbol recognition using on-line and off-line features». En: *IEEE International Conference on Acoustics, Speech, and Signal Processing* (1996), págs. 3438-3441.

[34] R. Yamamoto, S. Sako, T. Nishimoto y S. Sagayama. «On-Line Recognition of Handwritten Mathematical Expressions Based on Stroke-Based Stochastic Context-Free Grammar». En: *IEIC Technical Report* ().

[35] R. Zanibbi y D. Blostein. «Recognition and retrieval of mathematical expressions». En: *Int. J. Doc. Anal. Recognit.* (2012), págs. 331-357.

[36] R. Zanibbi, D. Blostein y J. Cordy. «Recognizing mathematical expressions using tree transformation». En: *IEEE Trans. Pattern Anal. Mach. Intell.* (2002), págs. 1-13.

[37] F. Álvaro, J.A. Sánchez y J.M. Benedí. «An integrated grammar based approach for mathematical expression recognition». En: *Pattern Recognition* (2016), 135-147.

[38] F. Álvaro, J.A. Sánchez y J.M. Benedí. «Classification of on-line mathematical symbols with hybrid features and recurrent neural networks». En: *International Conference on Document Analysis and Recognition* (2013), págs. 1012-1016.

[39] F. Álvaro, J.A. Sánchez y J.M. Benedí. «Recognition of on-line handwritten mathematical expressions using 2d stochastic context-free grammars and hidden Markov models». En: *Pattern Recognit. Lett. 35* (2014), págs. 58-67.

[40] F. Álvaro y R. Zanibbi. «A Shape-Based Layout Descriptor for Classifying Spatial Relationships in Handwritten Math». En: *ACM Symposium on Document Engineering* (2013), págs. 123-126.