UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Development of a scalable database for recognition of printed mathemematical expressions

DEGREE FINAL WORK

Degree in Computer Engineering

Author: Dan Anitei

Tutor: Joan Andreu Sánchez
José Miguel Benedí

Course 2019-2020

Resumen

Buscar información en documentos científicos impresos es un reto problemático que recientemente ha recibido atención especial por parte de la comunidad de investigación de Reconocimiento de Formas. Las Expresiones Matemáticas son elementos complejos que aparecen en documentos cientificos, y desarrollar técnicas para localizarlas y reconocerlas requiere preparar data sets que pueden ser utilizados como punto de referencia. La mayoría de las técnicas actuales para lidiar con Expresiones Matemáticas están basadas en técnicas de Reconocimiento de Formas y Aprendizaje Automático y por tanto, estos data sets tienen que ser preparados con información sobre el ground-truth para entrenamiento y test automático. Sin embargo, preparar data sets grandes es muy costoso y requiere mucho tiempo. Este proyecto introduce un data set de documentos científicos que ha sido preparado con el fin de reconocer y buscar Expresiones Matemáticas. Este data set ha sido generado automáticamente a partir de la versión LaTeX de los documentos y consecuentemente puede ser aumentado fácilmente. El ground-truth incluye la posición a nivel de página, la versión LaTeX de las Expresiones Matemáticas integradas y aisladas del texto y la secuencia de símbolos representados como *unicode code points* que se han utilizado para definir estas expresiones. En base a este data set, se han extraído estadísticas como por ejemplo el número total y el tipo de las expresiones, el número medio de expresiones por documento y las frecuencias de distribución de todo el conjunto de expresiones. En este documento también se introduce un experimento de clasificación de símbolos matemáticos que puede ser utilizado como punto de partida.

Palabras clave: LaTeX, aprendizaje automático, expresiones matemáticas, reconocimiento de formas, redes neuronales convolucionales

Abstract

Searching information in printed scientific documents is a challenging problem that has recently received special attention from the Pattern Recognition research community. Mathematical Expressions are complex elements that appear in scientific documents, and developing techniques for locating and recognizing them requires preparation of data sets that can be used as benchmarks. Most of the current techniques for dealing with Mathematical Expressions are based in Machine Intelligent techniques and therefore these data sets have to be prepared with ground-truth information for automatic training and testing. However preparing large data sets with ground-truth is a very expensive and time-consuming task. This project introduces a data set of scientific documents that has been prepared for Mathematical Expression recognition and searching. This data set has been automatically generated from the LaTeX version of the documents and consequently can be enlarged easily. The ground-truth includes the position at page level, the LaTeX version for Mathematical Expressions both embedded in the text and displayed and the sequence of mathematical symbols represented as unicode code points used to define these expressions. Based on this data set, statistics such as the total number and type of expressions, the average number of expressions per document and their frequency distribution were extracted. A baseline classification experiment with mathematical symbols from this data set is also reported in this paper.

Key words: data set, LaTeX, machine-learning, mathematical expressions, ground-truth, pattern recognition, convolutional neural networks

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Searching in documents related to science, technology, engineering and mathematics (STEM) is one of the most usual activities for researchers, scholars and scientists worldwide. Searching plain text in electronic STEM documents created in the digital era in massive collections is considered a solved problem. One of the challenges related to searching for information in STEM documents is to deal with complex structures like chemical formulas, plots, draws, maps, and mathematical expressions, among many others. This paper focuses on mathematical expressions (MEs) in electronic STEM documents [2].

1.1 Motivation

Searching for plain text in electronic documents is considered a solved issue, whether it involves digital images or editable documents. Searching in editable text can be done based on regular expressions, however, searching for MEs in large collections of digitally printed images still poses a significant challenge. One of the many challenges such a search poses relates to the method of typesetting a ME within a document. MEs can be either embedded along the lines of the plain text or isolated from the text. We refer to the first ones as *inline* MEs and to the second ones as *displayed* MEs.

Locating displayed MEs is a challenge already solved. Because displayed MEs are separated from the text, profile projection methods provide accurate results. Exceptions still occur since MEs are not the only non-text structures that can be included in texts, other examples including tables, figures, plots and other graphic elements. Developing a functional, universal method for locating inline MEs is still ongoing, since inline MEs can easily be confused with the surrounding text.

Current technology for searching MEs in documents is based on machine intelligent methods that need large amounts of training data [3, 4] with the necessary ground-truth. However, this ground-truth is usually prepared manually. In fact this is one of the bottle-necks for research on automatic methods. Developing techniques for preparing large data sets with ground-truth (GT) is a real need and one of the motivating factors behind this paper.

A second obstacle for inline ME identification, besides having access to large enough labeled data sets, refers to identifying ME symbol relationships. ME symbol classification is a well-studied problem that currently is not a main issue [5]. Letters contained in MEs can be interchangeable or not depending on whether they represent variables or not. As a result, MEs can be represented in several ways without changing their meaning. However, the relationship between the symbols of a ME is meaningful, posing thus a bigger challenge. Developing techniques that are able to capture these relationships and use them in a search engine is currently an interesting challenge [2]. This means that data sets that contain this information have to be created in order to develop a machine intelligent system than can be trained automatically. As a result, as secondary objective of this paper, we chose to create a data set of mathematical symbols encountered in the MEs extracted. The reason for creating this data set was to illustrate the potential uses of the main data set created. To further illustrate the potential, we conducted an experiment of classifying mathematical symbols that could serve as a precursor to a ME recognition system.

Besides using the ground-truth for ME symbol recognition, we also found it interesting to use extracted MEs for analyzing their occurrence frequency distribution. Being able to group MEs into clusters of similar expressions, opens the way to building a tolerant search engine within large collections of digital documents.

We chose to support the IBEM project by building a 200 STEM document, labeled data set with a rich ground-truth extracted automatically, accompanied by a secondary data set of mathematical symbols and characters extracted from the main data set, as well as by a ME frequency distribution analysis. The current paper provides both the resources for indexing and searching MEs on a large scale as well as the first stages of analysis of the ground-truth.

To be able to deliver the 200 document data set, we had to overcome several difficulties such as:

1. Copyright issues - many scientific documents are under some type of intellectual property rights protection form and parsing such data sets is either expensive or simply not possible.

2. The data set has to be large - the methods of writing a ME vary greatly based on the document's topic, field and area, on the author's preference, on the type of document (e.g. articles or slides), on the publisher's requirements, etc. As a result, the data set needs to be large enough to capture this variability.

3. The data set needs to be scalable - it should be possible to increase the data set in time and also enrich its GT with as much information as possible.

4. Automated processing - the data set should be optimized for automated processing.

Out of the issues presented, we consider the first one - copyright issues - as the most relevant, since all the other can-not be tackled without solving the first. To summarise, generating a large enough data set of STEM documents with GT, that is not affected by copyright infringements but is optimized for automated processing is a considerable challenge and the focus of this project.

On a more personal note, this project introduces an opportunity for me to learn how data sets are processed and created. Information extraction was a topic I have been interested in, and this project gives me a clearer perspective on how to tackle real life problems and convert them into information that can be used for building machine learning algorithms capable of improving automatically through experience.

Having said that, this paper introduces a data set of printed scientific documents for ME searching and recognition research. The data set has been prepared automatically from a public set of documents. The preparation of the GT for each document is carried out from the LaTeXversion. This makes this data set scalable to include thousands of documents that can be used for researching and developing efficient searching techniques. This paper also includes a baseline classification experiment with mathematical symbols from this data set.

1.2 Objectives

The main objective of this project is to create a data set of 200 STEM documents with a rich ground-truth containing information about the position of each ME and their LaTeX transcript.

In Machine Learning related literature, ground-truth refers to data characteristics that are important for algorithm training. The ground-truth for this project refers to characteristics that we considered that would be of importance for giving a solution to the challenge of searching for MEs in large collections of digitally printed images. The ground-truth would serve as a baseline for measuring the accuracy of algorithms that provide a solution for this challenge.

The main objective has been divided into several specific objectives listed below:

- Automatically process 1 000 STEM documents from a publicly available data set in order to extract the ground-truth. Out of these 1 000 documents, the ground-truth of 200 documents will be visually validated in order to create the primary data set.

The core ground-truth for these documents would include: the ME location (page number and page coordinates), the ME LaTeX transcript, the type of ME (inline or displayed), as well as several other characteristics.

- Develop a software system with an accuracy of at least 20%, capable of processing LaTeX files and correctly extract the ground-truth

The software system would be capable of extracting the core ground-truth mentioned above. The results obtained by the software system would have to be manually validated. For this reason, we choose a volume of 200 documents. Validation of the processed documents is conducted by visually checking that the highlighted bounding boxes are in the correct position and have the correct dimensions. As a result, the process of validation is very time consuming. In the future our aim is to reach a volume of 600 documents, but for this project we consider that reaching a data set of 200 documents is a very good result.

- Conduct an experiment that generates and analyzes a secondary data set of images of all the symbols that define the MEs extracted from the primary data set.

Starting from the core ground-truth that includes the ME LaTeX transcript, image representations of each symbol shall be extracted, with the purpose of labelling, classifying and analyzing each symbol.

- Conduct an experiment that analyzes the MEs and calculates their occurrence frequency distribution.

Both experiments aim to illustrate the potential uses of the primary dataset obtained. The further use of the resulting database is explored under chapter 7, as well as several ways to improve or adapt the ME identification methodology.

1.3 Paper Structure

This paper has been divided in several sections in order to facilitate the assessment of the degree of fulfilment of the objectives set for this project, and to provide a clear guide of the steps needed for replicating the results presented.

After an initial analysis of the motivation behind this project in chapter 1 and the related work in chapter 2, this paper presents the design of the IBEM data set (chapter 3) starting with data collection and the process of establishing features of the data set that would be of interest when creating the ground-truth.

In chapter 4 we detail the process of extracting the ground-truth and the validation phase, after which we present a secondary data set with the purpose of enriching the ground-truth with information about the mathematical symbols used to define the MEs. In this chapter we also perform an analysis of the MEs by carrying out a clustering experiment in order to group together similar expressions and calculate their frequency distribution.

The structure of the IBEM data set is presented in chapter 5 where we detail all the information (textual or visual) presented in the data set as it will be made publicly available.

Once the structure of the data set is presented, the paper introduces a mathematical symbol classification experiment (chapter 6) based on the information extracted from processing MEs. This experiment consists of designing a classification model, presenting the data used to train it, and the results obtained from testing the model.

Potential future uses of the data set are analyzed in chapter 7. This chapter also explores some improvements that can be brought to the ME highlighting module presented in the paper.

Finally, this paper concludes with chapter 8 with a summary of the key points of this project. In addition, we provide a description of the architecture of the software system presented in this paper and how to launch the project into execution (Appendix A), and also present some of the most important regular expressions used for automating the processing of the ground-truth (Appendix B).

CHAPTER 2
Related Work

There are several data sets of typeset MEs that have been used in the past for several researches. The UW-III data set [6] is a well-known data set but the amount of data is not very large. It consists of 1600 scanned images of English documents for which the ground-truth was manually edited. The GT can be used for symbol classification and the LaTeXversion is also available.

In this data set, various bounding boxes have been inserted for each image, highlighting amongst others text and non-text zones, lines, words, page frames and other. It is important to note that the quality of the scans was not consistent throughout the data set. The scans were not always done based on the original documents, but sometimes on copies of copies (to the nth degree) were included, as well as images where the scan of a page included sections of the additional next page. While this situation proved useful for several research projects [7], using UW-III for ME detection would not have yielded optimum results. First of all, the scans would have had to be processed using OCR technology to enable ME identification. This step would have had limited success, as OCR technology is directly dependent on high quality scans for correct interpretation of each character and even with high quality input it does not always relay correct results. Secondly, inserts of the next page into the same image as the processed page would have led to situations where MEs are detected on the surplus section and then again when processing the next page.

Another data set is the InftyCDB-1 data set [8]. This data set and its later versions were developed in the Infty project [1]. The Infty project - **Research Project on Mathematical Information Processing** - is often referenced in papers related to identification of mathematical expressions in digital documents. Though the project covers the entire process of building a data set, as well as building and training an algorithm that can extract MEs from OCR-ed documents, for the purpose of this paper, it is interesting to analyze the steps took for building the data set.

The data set is available in four different versions, shown in Table 2.1. Since the project aims to correctly identify the characters in mathematical expressions from scanned documents, when building the data set, a limited number of articles were selected.

The approach used to build the data sets shown in Table 2.1 included scanning documents at 400 dpi and "depending on when it was selected, a symbol may have been scanned into a gray image and then converted into binary images using different thresholds to produce character images of different density from one original sample, while some others are scanned directly into binary image from scanner using medium threshold" [2]. One of the drawbacks this data set is that the ground-truth is extracted from articles that are not copyright free. For this reason these articles are not provided with the data sets, making the testing and comparing of systems more complicated. Also, this data set does not include matrices, tables and figures and the relationship among symbols in a ME was defined manually, and the markup language is not included in the GT.

Another important data set is the IM2LATEX-100K [3]. This data set has 103 556 different LaTeX MEs along with rendered pictures. The MEs were extracted by parsing LaTeX sources of papers from tasks I and II of the 2003 KDD cup [9]. The problem that we identified in this data set is that it is usefull for researching on MEs parsing but not for the detection of MEs in the context of the article they come from. This issue is pointed in [4], and therefore they proposed a new data set that contains 47 articles with 887 pages, but the total number of MEs is not

[1]http://www.inftyproject.org/en/index.html

[2]http://www.inftyproject.org/download/AboutInftyCDB-3_en.txt

data set	**Number of articles**	**Number of MEs and pages in selected documents**
InftyCDB-1 – A Ground Truth Database of Characters, Symbols words and Formulas in Mathematical Documents; First Distribution, March 18, 2005	30 articles in English	21 056 mathematical expressions, 476 pages.
InftyCDB-2 – A Ground Truth Database of Characters, Symbols, words and Formulas in Mathematical Documents; Second Distribution, December 27, 2006	26 articles in English, 4 articles in French and 7 articles in German	21 056 mathematical expressions. number of pages n/a
InftyCDB-3 – A Ground Truth Database of Characters, Symbols in Mathematical Documents; Third Distribution, October 2006	20 articles (a subset of those used in InftyCDB-1)	mathematical expression structure is not included, 346 pages.
Infty-MDB-1 – A Ground Truth Database of Mathematical Expressions, August 12, 2009	32 articles	4 400 mathematical expressions number of pages n/a

Table 2.1: Infty project data set.

provided. Finally, it is worth mentioning that this last data set has been used in a competition on MEs detection [2].

All these limitations make necessary the development of a data set that overcome these problems: thousands of images of pages from scientific documents, with MEs located and annotated, with the markup language available. This data set will be made available for research purposes.

CHAPTER 3

Design of IBEM data set

3.1 Data Collection

As stated previously in section 1.1 of the introduction, the data set chosen for the purposes of this paper had to meet several criteria: it had to be publicly available, and contain a large number of STEM documents written in LaTeX in order to facilitate the automation of the extraction of the ground-truth. For this reason we chose the KDD Cup data set [9]. This data set is well known and is being used for knowledge discovery and data mining purposes. More importantly, this collection of documents is publicly available and it allowed us to overcome copyright issues.

The KDD Cup data set is a large collection of research papers from the year 1992 until 2003 inclusive, with approximately 29 000 documents in total with 1.7 gigs of data. The LaTeX sources of all papers are available for downloading. These papers are indexed by the publishing year and have been assigned a unique random paper id between 1 and 100 000.

As the KDD Cup contains papers published from 1992 until 2003, we started selecting papers from the year 2000 onward to avoid compatibility issues with older versions of libraries potentially used by the authors.

In addition to selecting the papers according to the year, we removed those documents that did not compile with the 2019 version of `texlive` (TeX 3.14159265).

This compilation error was due to obsolete versions or missing auxiliary files given that only the main LaTeX source was provided.

Table 3.1 shows the resulting number of documents after these two filters.

Table 3.1: number of documents

Total no. of documents	29 556
No. of documents since 2000	10 611
No. of documents that compiled correctly	2 791

3.2 Ground-Truth

Having access to the LaTeX sources of these papers would provide us a good way of automating the extraction of the ground-truth. We developed a set of preliminary regular expressions (regex) that would detect the tags of these delimiters and create statistics for the documents that compiled correctly to give us an idea of the potential of this data set.

One of the characteristics of writing equations in LaTeX is using the full name of the delimiters provided by LaTeXfor enclosing the expressions in a mathematical environment. There is an alternative solution to this by defining macros to abbreviate the formal notation and make writing easier for the authors. By renaming the LaTeX standard delimiters and separating them from the definition of MEs, the complexity of the process of detection would increase. We noticed that many authors preferred this latter approach. Considering that we used regular expressions to insert LaTeX commands for extracting the ME, the situation raised syntax problems too complex to solve by means of regular expressions. Out of a total of 2 791 documents approximately 1 000 documents did not rename the delimiters of the mathematical environments, therefore we decided to focus only on these documents. Table 3.2 indicates the characteristics of this collection of documents.

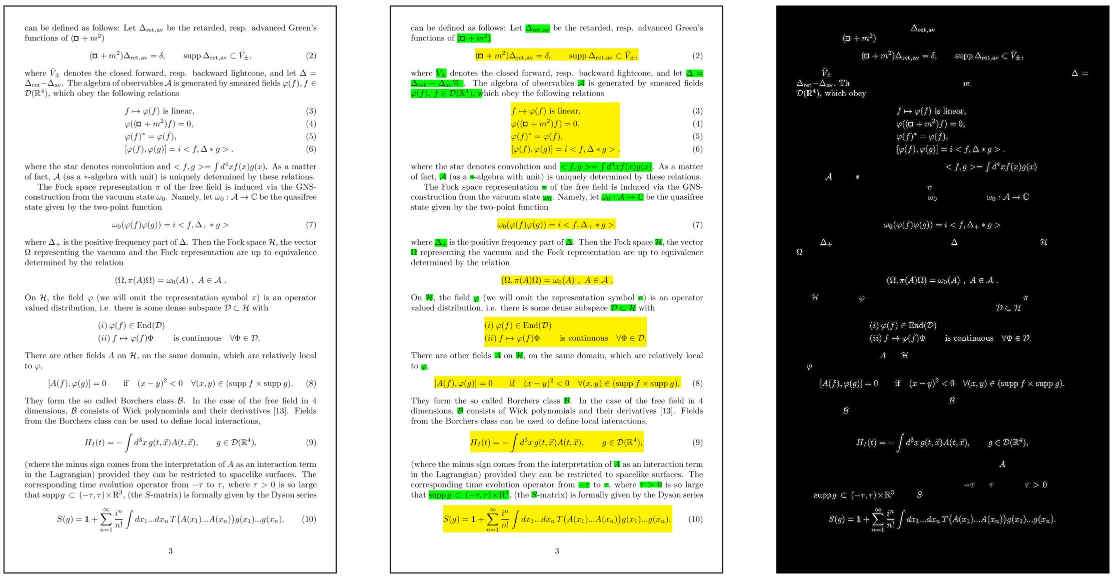
Out of these documents we extracted as much information as possible. This ground-truth would contain information about the position of the MEs, their type and LaTeX transcript. We also decided to highlight the bounding boxes of the MEs

Table 3.2: statistics about the collection of papers.

Total no. of documents	957
Total no. of pages	14 408
No. of displayed MEs	102 921
No. of inline MEs	450 339
Average no. of pages per document	15.06
Average no. of displayed MEs per document	107.55
Average no. of inline MEs per document	470.57

which would allow us to create color inverted images of the pages with only the definition of the MEs. Highlighting the MEs would also provide a way of visually verify and validate the ground-truth.

An example of the output described before can be seen in Figure 3.1.



can be defined as follows: Let $\Delta_{\mathrm{ret,av}}$ be the retarded, resp. advanced Green's functions of $(\square + m^2)$

$$(\square + m^2)\Delta_{\mathrm{ret,av}} = \delta, \qquad \mathrm{supp}\,\Delta_{\mathrm{ret,av}} \subset \bar{V}_\pm, \tag{2}$$

where $\bar{V}_\pm$ denotes the closed forward, resp. backward lightcone, and let $\Delta = \Delta_{\mathrm{ret}} - \Delta_{\mathrm{av}}$. The algebra of observables $\mathcal{A}$ is generated by smeared fields $\varphi(f)$, $f \in \mathcal{D}(\mathbb{R}^4)$, which obey the following relations

$$f \mapsto \varphi(f) \text{ is linear,} \tag{3}$$
$$\varphi((\square + m^2)f) = 0, \tag{4}$$
$$\varphi(f)^* = \varphi(\bar{f}), \tag{5}$$
$$[\varphi(f), \varphi(g)] = i < f, \Delta * g > . \tag{6}$$

where the star denotes convolution and $< f, g >= \int d^4x f(x) g(x)$. As a matter of fact, $\mathcal{A}$ (as a $*$-algebra with unit) is uniquely determined by these relations.

The Fock space representation π of the free field is induced via the GNS-construction from the vacuum state ω_0. Namely, let $\omega_0 : \mathcal{A} \to \mathbb{C}$ be the quasifree state given by the two-point function

$$\omega_0(\varphi(f)\varphi(g)) = i < f, \Delta_+ * g > \tag{7}$$

where Δ_+ is the positive frequency part of Δ. Then the Fock space $\mathcal{H}$, the vector Ω representing the vacuum and the Fock representation are up to equivalence determined by the relation

$$(\Omega, \pi(A)\Omega) = \omega_0(A)\ ,\ A \in \mathcal{A}\ .$$

On $\mathcal{H}$, the field φ (we will omit the representation symbol π) is an operator valued distribution, i.e. there is some dense subspace $\mathcal{D} \subset \mathcal{H}$ with

$$(i)\ \varphi(f) \in \mathrm{End}(\mathcal{D})$$
$$(ii)\ f \mapsto \varphi(f)\Phi \qquad \text{is continuous} \quad \forall \Phi \in \mathcal{D}.$$

There are other fields A on $\mathcal{H}$, on the same domain, which are relatively local to φ,

$$[A(f), \varphi(g)] = 0 \qquad \text{if} \quad (x-y)^2 < 0 \quad \forall (x,y) \in (\mathrm{supp}\, f \times \mathrm{supp}\, g). \tag{8}$$

They form the so called Borchers class $\mathcal{B}$. In the case of the free field in 4 dimensions, $\mathcal{B}$ consists of Wick polynomials and their derivatives [13]. Fields from the Borchers class can be used to define local interactions,

$$H_I(t) = -\int d^3x\, g(t, \vec{x}) A(t, \vec{x}), \qquad g \in \mathcal{D}(\mathbb{R}^4), \tag{9}$$

(where the minus sign comes from the interpretation of A as an interaction term in the Lagrangian) provided they can be restricted to spacelike surfaces. The corresponding time evolution operator from $-\tau$ to τ, where $\tau > 0$ is so large that $\mathrm{supp} g \subset (-\tau, \tau) \times \mathrm{R}^3$, (the S-matrix) is formally given by the Dyson series

$$S(g) = \mathbf{1} + \sum_{n=1}^{\infty} \frac{i^n}{n!} \int dx_1 ... dx_n\, T\big(A(x_1)...A(x_n)\big) g(x_1)...g(x_n). \tag{10}$$

3

(a) Original page. **(b)** Highlighted BBs. **(c)** Color inverted.

Figure 3.1: output obtained by processing page three of paper 0001129 [1].

The GT obtained from this data set is generated in different formats:

- Jpeg images corresponding to each page of the original and the modified pdf format, in which MEs are enclosed in a bounding box highlighted in color. While images of the original pages can directly be used for testing ML algorithms, pages of the modified pdf format are included for visually checking the ground-truth used for training and could prove useful when evaluating these algorithms.
- Color inverted images of each page containing only the definition of the MEs. These color inverted images can be used as binary masks that can prove very useful for ME extraction.
- A text file containing all the inline MEs detected per document. Useful for analyzing the definitions of inline MEs and perform experiments like the ones presented in section 4.4 and chapter 6.
- A text file containing all the displayed MEs detected per document. Useful for analyzing the definitions of displayed MEs. Useful for analyzing the definitions of displayed MEs and perform experiments like the ones presented in section 4.4 and chapter 6.
- A text file containing the coordinates of the bounding boxes enclosing the MEs. Useful for training algorithms for detecting and extracting MEs.

CHAPTER 4

Preparing the data set

4.1 Extracting the Ground-Truth

As one of the most important parts of this paper, this section presents the process of extracting the ground-truth from LaTeX STEM documents, with the challenges that arose during implementation and the solutions given to these challenges. In order to develop a software system capable of processing LaTeX files and automatically extract the GT, we used a mixture of techniques such as regular expressions, LaTeX macro programming and computer vision.

Most of the challenges we faced were related to the use of regular expressions, while some were due to the flexibility of LaTeX typesetting and the high variability in the definition of MEs expected when working with large collections of documents. These are the most important challenges we faced, and their solutions, organized in ascending order by difficulty:

- The data set included a large number of LaTeX mathematical environments that is sure to increase in time. Given that MEs are searched for and detected through regular expressions, the list of mathematical environments needs to be updated and maintained in order to assure that all types of MEs are processed.

- In LaTeX there are two types of MEs, inline and displayed. Inline expressions are embedded along the lines of the plain text while displayed expressions are isolated from it. Given that these two type of MEs are rendered accordingly to the surrounding elements, we had to treat each type separately.

- It was essential that the original structure of the documents does not suffer modifications when inserting LaTeX macros for extracting and highlighting the definitions of MEs. Given that in LaTeX, macros can contain both commands and typeset characters, it is very easy to insert unwanted blank spaces when defining these macros. The solution to this problem was to compare the appearance of the documents before and after introducing any new macros. There are tools that compare pdf formats by overlaying one document on top of each other highlighting the differences.

- The definition of MEs can run in more than a single line, making the correct detection of the expression more complex. This challenge arose due to the way in which text stream editors work by processing the document one line at a time for matching regular expressions. One solution to this problem was to append multiple lines in the search buffer. This behaviour introduced a new problem of incorrectly joining the definition of MEs if defined close to each other. This was solved by using a neutral character like the commentary symbol that we inserted after each ME definition in order to signal the ending of such definition. This neutral character was later removed so as not to introduce any unwanted behaviours.

- Highlighting the bounding boxes of MEs introduced several difficulties due to the geometrical position of MEs. We had to take into account all the special cases that could arise given the high variability in the typesetting of MEs. The solution to this problem was to increase the number of coordinate measurements in order to better approximate the position and dimension of the bounding boxes. This greatly increased the complexity of the solution which favoured imprecisions in the highlighting process.

- Some parts of the ground-truth, such as the page number of MEs, could not be accurately calculated within LaTeX, due to inconsistencies in the shipout routine of LaTeX that could suffer changes after the highlighting and extraction phase. Therefore, the ground-truth had to be extracted after finalizing the renderization of these documents, which would guarantee that the ground-truth would not suffer any other modifications. This was done using computer vision techniques for performing shape analysis on the resulting highlighted MEs.

All of the above problems and their solutions will be explored with greater detail in this section.

As explained previously, there are two types of MEs in LaTeX, inline and displayed, enclosed in special math delimiters. Inline equations can be defined by using the `$...$` or `\(...\)` delimiters. In the case of displayed equations the usual delimiters are `$$...$$`, `\[...\]` and `\begin{mathEnv}...\end{mathEnv}`, where "..." denotes the definition of the equation and `mathEnv` stands for mathematical environments like the `eqnarray`, `align`, `equation`, `gather`, `multline`, `displaymath` environments among others. All these delimiters are available by default in LaTeX or by importing libraries such as `Amsmath` [10].

Even though the use of the `eqnarray` environment is not recommended [11], there are papers in this data set making use of this environment, and we decided not to modify them so as not to change the original structure of these documents. While modifying the original documents would affect the reproducibility of this project and of the results presented in this paper, changing the structure of the documents resulted from processing the original files would render the ground-truth obtained not usable. Any unwanted shift on the x or y axis of the bounding boxes of MEs would affect the accuracy and quality of the ground-truth.

In order to obtain the ground-truth and the output shown in Figure 3.1, we divided the process in two parts. The first part consisted in creating LaTeX macros for highlighting and extracting MEs which would be inserted into the documents by means of regular expressions created in the second part.

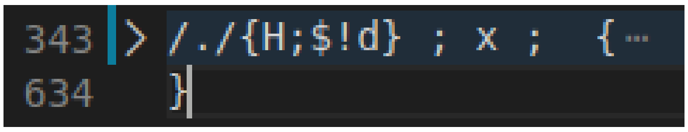
The extraction of MEs was done by creating LaTeX macros that would write the definition of MEs to a file differentiating between inline and displayed expressions. In this case, the regular expressions designed had the sole purpose of automating the process by detecting the tags of the delimiters presented before and inserting the macros we created. For this reason we decided to only focus on documents that did not rename these tags. Examples of the regular expressions used for inserting LATEX macros are shown in Appendix B.

Since the number of regular expressions would be quite large given the many alternatives of environments that LATEX provides for defining MEs, in this project we used SED [12] as the preferred text stream editor considering that SED works by making only one pass over the input, thus making it very efficient.

Special care was needed to create the regular expressions used to insert these macros, since the starting and ending delimiters were usually not on the same line and knowing that SED processes the text one line at a time. In order to give solution to this problem, we took advantage of the fact that SED uses two buffers (pattern buffer and hold buffer) for the matching process. The hold buffer is an auxiliary buffer that SED allows access to and is used to save all or part of the SED pattern space for future retrieval. Information can be appended/swapped between the pattern and the hold space by using a set of functions:

- *H*: Append a newline to the contents of the hold space, and then append the contents of the pattern space to that of the hold space.
- *x*: Exchange the contents of the hold and pattern spaces.

With these functions we were able to create a SED command for appending consecutive lines into the hold pattern until an empty line would appear, thus making possible to search patterns in paragraphs instead of having to restrict the search to only individual lines. Figure 4.1 shows the code snippet of the command used.



```
> /./{H;$!d} ; x ;  {⋯
  }
```

Figure 4.1: command used for searching patterns in paragraphs.

Another characteristic of text stream editors is that they function in a greedy way as explained in [12]: *"Note that the regular expression matcher is greedy, i.e., matches are attempted from left to right and, if two or more matches are possible starting at the same character, it selects the longest"*. This behaviour could incorrectly join the expression of two or more MEs if defined close to each other. This posed a problem since we previously changed the normal pattern matching mechanism of SED by now searching in paragraphs instead of just individual lines. To solve this, our approach had been broken down into these following steps:

1. Firstly, we eliminated all the comments written in the document. Eliminating comments had no effect on the rendered version of the text, as comments are not shown in the output. We did not remove lines containing only the comment symbol (`%`), as these lines served the purpose of suppressing unwanted space resulting from line breaks.
2. Once all comments were removed, the next step was to insert the `%` symbol after the definition of every ME.
3. Given that `SED` has a greedy behaviour, when creating the regular expressions for detecting MEs we made sure that SED's regular expression matcher would search for the delimiters presented before and match as many characters possible as it could without including the `%` symbol, thus avoiding pairing MEs defined successively.
4. Lastly, we removed the trailing `%` symbols as to not introduce unwanted behaviour when compiling the LaTeX source of the documents.

Highlighting the bounding boxes of MEs was more complex as we had to take into account some special cases that we will discuss below. First of all, we had to calculate the position of the MEs as rendered on the page. It is important to remark that detecting the geometrical position of a ME introduces several difficulties such as:

- An inline ME can run in more than one single line.
- An inline ME can run in more than one single line, and in different pages.
- An inline ME can run in more than one single line, and in different pages, and/or in different columns.
- An inline ME can appear in a caption, in a footnote, in a draw, or in a plot.

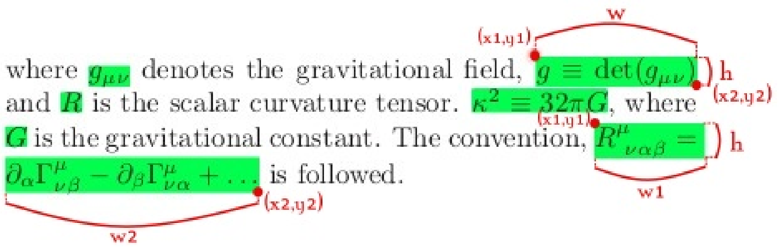
- A displayed ME can run in more than one page and/or column.
- A displayed ME can include a numbering.

These situations had to be taken into account when processing the data set.

For each type of MEs, inline or displayed, we used different approaches to compute the location of the bounding boxes. We made use of the module `savepos` provided by the package `zref` [13] to get the absolute coordinates of the starting and ending points of the expressions as rendered on the page. Once these coordinates were calculated, the dimensions of the bounding box were computed by measuring the rendered definition of the expression. It is important to know that we compiled each LaTeX source three times. Twice to get the correct coordinates and a third in order to compute the bounding boxes using these coordinates. The third compilation was necessary since every expression is processed in a sequential manner and it was impossible to use the ending coordinate if it was not previously calculated. With these dimensions and with the absolute position of each bounding box determined we were able to highlight the expressions.

In the case of inline MEs, it was sufficient to only take two coordinates for the starting and ending point of the expressions. By comparing these coordinates we deducted if these expressions were rendered on one line or more, or even if they were split over two pages.

Highlighting one-lined expressions is trivial and can be done directly by enclosing the definition of these MEs in a colored box or by drawing a box over the background, taking into account that we previously computed both the location and the dimensions of the MEs. This was not the case for expressions rendered on two or more lines, because enclosing the definition in a colored box, would force the ME to be rendered on one line not allowing for line breaks or page breaks and therefore would modify the original structure of the document. In this case we used the package `tikz` [14] to be able to draw directly on the background of the page. The first line of the expression was highlighted by drawing a colored rectangle from the first coordinate to the end of the line. Between the first and last line, in case the expression was rendered on more than two lines, we drew line long rectangles by looping until we reached just above the ending coordinate. The last line was highlighted in similar fashion as the first one, by drawing a rectangle from the beginning of the line to the ending coordinate of the expression. Figure 4.2 shows how to determine if an inline ME is split over more than one line.

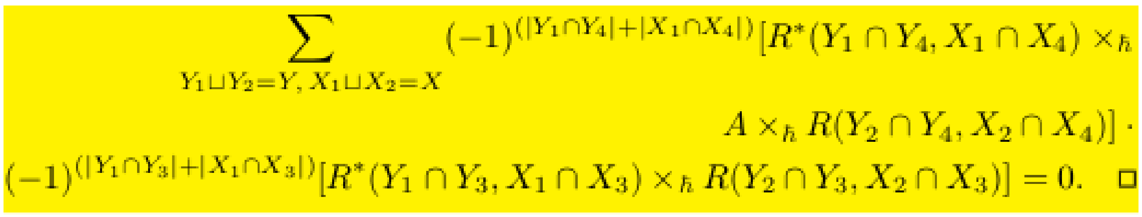


where $g_{\mu\nu}$ denotes the gravitational field, $g \equiv \det(g_{\mu\nu})$ and R is the scalar curvature tensor. $\kappa^2 \equiv 32\pi G$, where G is the gravitational constant. The convention, $R^{\mu}{}_{\nu\alpha\beta} = \partial_\alpha \Gamma^{\mu}_{\nu\beta} - \partial_\beta \Gamma^{\mu}_{\nu\alpha} + \ldots$ is followed.

Figure 4.2: inline mathematical expression running over two lines.

The height and width of ME are calculated within LaTeX by making a copy of the expression and enclosing that expression in a paragraph box that can be measured directly. In Figure 4.2 it can be observed that by checking if the difference between the y components is greater than the height of the line, then the expression is rendered on more than one line. By dividing that difference between the height of the line, we can calculate over how many lines the expression is split.

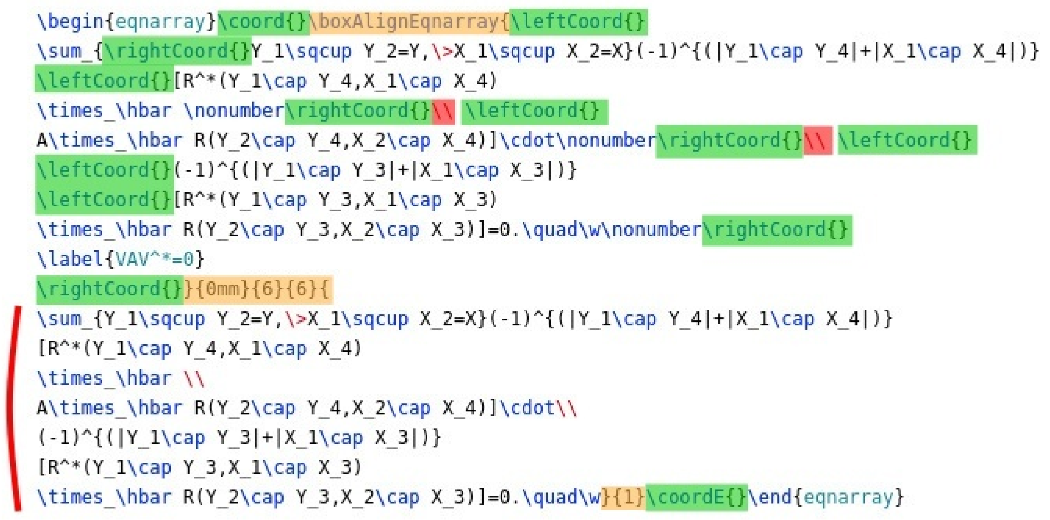
In the case of displayed MEs, taking the beginning and ending coordinates to compute the upper left and lower right corner of the bounding box did not guarantee that the first and last symbols of the expression were rendered as extreme points. Figure 4.3 shows an example of this situation.



$$\sum_{Y_1 \sqcup Y_2 = Y,\, X_1 \sqcup X_2 = X} (-1)^{(|Y_1 \cap Y_4| + |X_1 \cap X_4|)} [R^*(Y_1 \cap Y_4, X_1 \cap X_4) \times_\hbar A \times_\hbar R(Y_2 \cap Y_4, X_2 \cap X_4)] \cdot (-1)^{(|Y_1 \cap Y_3| + |X_1 \cap X_3|)} [R^*(Y_1 \cap Y_3, X_1 \cap X_3) \times_\hbar R(Y_2 \cap Y_3, X_2 \cap X_3)] = 0. \quad \square$$

Figure 4.3: example of a displayed ME rendered on page 18 of paper 0001129 [1].

In order to take into account situations like the one in Figure 4.3, macros were inserted to take coordinates before and after each newline symbol and also between the symbols of some mathematical elements such as fractions, sums, products, integrals, etc., which could have superscript or subscript elements making them be rendered in an upper or lower position than the first or last symbol, respectively. Figure 4.4 shows some of the key points in the definition of a ME where coordinate macros were inserted.



```
\begin{eqnarray}\coord{}\boxAlignEqnarray{\leftCoord{}
\sum_{\rightCoord{}Y_1\sqcup Y_2=Y,\>X_1\sqcup X_2=X}(-1)^{(|Y_1\cap Y_4|+|X_1\cap X_4|)}
\leftCoord{}[R^*(Y_1\cap Y_4,X_1\cap X_4)
\times_\hbar \nonumber\rightCoord{}\\ \leftCoord{}
A\times_\hbar R(Y_2\cap Y_4,X_2\cap X_4)]\cdot\nonumber\rightCoord{}\\ \leftCoord{}
\leftCoord{}(-1)^{(|Y_1\cap Y_3|+|X_1\cap X_3|)}
\leftCoord{}[R^*(Y_1\cap Y_3,X_1\cap X_3)
\times_\hbar R(Y_2\cap Y_3,X_2\cap X_3)]=0.\quad\w\nonumber\rightCoord{}
\label{VAV^*=0}
\rightCoord{}}{0mm}{6}{6}{
\sum_{Y_1\sqcup Y_2=Y,\>X_1\sqcup X_2=X}(-1)^{(|Y_1\cap Y_4|+|X_1\cap X_4|)}
[R^*(Y_1\cap Y_4,X_1\cap X_4)
\times_\hbar \\
A\times_\hbar R(Y_2\cap Y_4,X_2\cap X_4)]\cdot\\
(-1)^{(|Y_1\cap Y_3|+|X_1\cap X_3|)}
[R^*(Y_1\cap Y_3,X_1\cap X_3)
\times_\hbar R(Y_2\cap Y_3,X_2\cap X_3)]=0.\quad\w}{1}\coordE{}\end{eqnarray}
```

Figure 4.4: LaTeX transcript of the displayed ME shown in Figure 4.3.

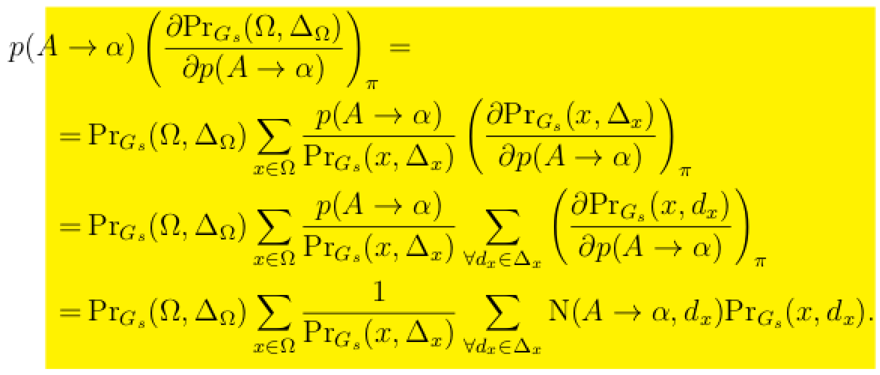
Considering that LaTeX macros are quite verbose which would make the regular expressions unnecessary long, instead of inserting the LaTeX code of the macros, we created a package file that would work as a library with the definition of these macros. In Figure 4.4 it can be seen that the ME is defined inside an `eqnarray` mathematical environment. An example of a regex for inserting coordinate macros for this expression, would be to search for `\begin{eqnarray}` and immediately add after, the call to the `\coord{}` macro that would take the starting coordinate of the ME. It is important to notice that the starting (`\coord{}`) and ending coordinates (`\coordE{}`) are inserted inside the `eqnarray` environment, because LaTeX treats the environment as a paragraph adding padding space around it to separate it from the adjacent paragraphs. Inserting the coordinates on the outside of the environment would shift the coordinates to take into account this added blank space and the bounding box would be measured incorrectly.

In Figure 4.4 all coordinate macros are colored with green. It can be seen that such macros were inserted before and after the newline symbol \\ marked with a red box. It can also be seen that in the second line of the definition of the ME, we inserted a coordinate macro just after opening the \sum expression. As explained before, sums, products or integral expressions, can have superscript or subscript elements that would be rendered at a different height than the rest of the symbols. Given that such expressions follow a syntax like `\sum_{lower part}^{upper part}`, we inserted the macro in that position in order to be taken into account when computing the lower rightmost corner of the ME. Given that this instance of the `\sum` expression does not have an upper part, no `\leftCoord{}` macro was inserted in the definition in the definition of the sum.

At the same time we inserted the call for the starting and ending coordinates, we also inserted the call for the macro `\boxAlignEqnarray{...}` that is colored yellow in Figure 4.4. This macro receives 6 arguments such as: 1) the definition of the ME; 2) a negative shift to be applied if necessary; 3) and 4) the number of coordinates to take into account when computing the upper left and lower right corners of the bounding box; 5) a copy of the definition of the ME (marked in Figure 4.4 with a red open parenthesis) that would be used for measuring the dimensions of the bounding box; and 6) a boolean value that would be set to 1 if the environment in question is eqnarray. This parameter exists because the dimensions of eqnarray environments are a bit more trickier to measure as a result of the alignment between elements of the expressions.

When highlighting displayed MEs, as in the case of inline MEs, we had to take into account that the equation could be split over two pages or columns. This was done by processing the coordinate points in sequential order until there was a significant difference in the y coordinate which would imply a vertical gap in the definition of the ME and the need to treat each part of the expression independently.

A special case we encountered that was briefly mentioned before, was the use of negative spaces declared in the definition of MEs. This negative space would shift the position of the expression as rendered on the page and would create a misalignment in the highlighting of the bounding box. Vertical negative space would not pose problems, since such space would generally be declared above the definition of the expression in order to shift the whole expression upwards and reduce the space between it and the paragraph above. This negative space would be processed by LaTeX before reaching the coordinate macros we inserted and the computation of the position of the bounding box would not be affected by it. This was not the case for horizontal negative space that would be declared between symbols of the expression, and which could shift parts of the expressions outside of the highlighted bounding box if such space is not taken into account. Figure 4.5 is an example of such a case.



$$\begin{aligned} p(A \to \alpha) \left(\frac{\partial \mathrm{Pr}_{G_s}(\Omega, \Delta_\Omega)}{\partial p(A \to \alpha)} \right)_\pi &= \\ &= \mathrm{Pr}_{G_s}(\Omega, \Delta_\Omega) \sum_{x \in \Omega} \frac{p(A \to \alpha)}{\mathrm{Pr}_{G_s}(x, \Delta_x)} \left(\frac{\partial \mathrm{Pr}_{G_s}(x, \Delta_x)}{\partial p(A \to \alpha)} \right)_\pi \\ &= \mathrm{Pr}_{G_s}(\Omega, \Delta_\Omega) \sum_{x \in \Omega} \frac{p(A \to \alpha)}{\mathrm{Pr}_{G_s}(x, \Delta_x)} \sum_{\forall d_x \in \Delta_x} \left(\frac{\partial \mathrm{Pr}_{G_s}(x, d_x)}{\partial p(A \to \alpha)} \right)_\pi \\ &= \mathrm{Pr}_{G_s}(\Omega, \Delta_\Omega) \sum_{x \in \Omega} \frac{1}{\mathrm{Pr}_{G_s}(x, \Delta_x)} \sum_{\forall d_x \in \Delta_x} \mathrm{N}(A \to \alpha, d_x) \mathrm{Pr}_{G_s}(x, d_x). \end{aligned}$$

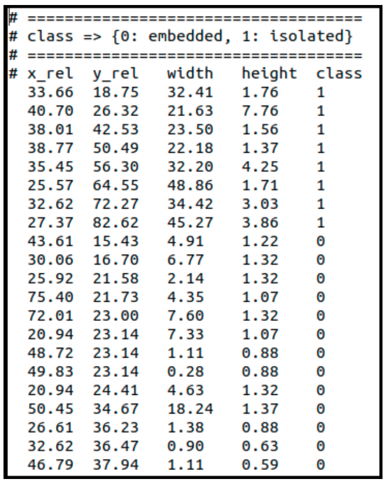
Figure 4.5: example of the misalignment in the highlighting process caused by a horizontal negative space.

The solution to this problem was to design a regular expression that would detect if any negative space was declared in the definition of the ME, and if it was the case pass the value as an argument to the macro that computes the bounding box taking the shift into account.

Once the MEs were highlighted (green for inline, yellow for displayed), we compiled the resulting LaTeX file to obtain a pdf format and convert each pdf page into an image for each page. Each such image was later processed[1] and passed through an yellow and green color filter in order to remove the running text and focus only on the definitions of the MEs. The output of this process was a negative similar to the one that is shown in Figure 3.1.

By performing a shape analysis on this negative we directly obtained relative coordinates of each ME by detecting the contour of these expressions. Figure 4.6 shows some of the coordinates obtained by processing the example shown in Figure 3.1.

[1] https://opencv.org/



```
# ===================================
# class => {0: embedded, 1: isolated}
# ===================================
# x_rel  y_rel   width   height  class
  33.66  18.75   32.41   1.76    1
  40.70  26.32   21.63   7.76    1
  38.01  42.53   23.50   1.56    1
  38.77  50.49   22.18   1.37    1
  35.45  56.30   32.20   4.25    1
  25.57  64.55   48.86   1.71    1
  32.62  72.27   34.42   3.03    1
  27.37  82.62   45.27   3.86    1
  43.61  15.43   4.91    1.22    0
  30.06  16.70   6.77    1.32    0
  25.92  21.58   2.14    1.32    0
  75.40  21.73   4.35    1.07    0
  72.01  23.00   7.60    1.32    0
  20.94  23.14   7.33    1.07    0
  48.72  23.14   1.11    0.88    0
  49.83  23.14   0.28    0.88    0
  20.94  24.41   4.63    1.32    0
  50.45  34.67   18.24   1.37    0
  26.61  36.23   1.38    0.88    0
  32.62  36.47   0.90    0.63    0
  46.79  37.94   1.11    0.59    0
```

Figure 4.6: example of some of the coordinates obtained by processing page three of paper 0001129 [1].

The coordinates shown in Figure 4.6 could have been obtained directly from within LaTeX since we already had to compute these coordinates in order to highlight the MEs. The problem with this approach was that after the highlighting phase, the page could still be further modified before being shipped-out and the coordinates calculated before would not be valid even if the highlighted output of the page would be correct. For this reason we chose to process these pages after the renderization process would finalize. A second reason was that by post-processing the documents, the color inverted images shown in Figure 3.1 were obtained as a by-product of calculating the coordinates, which would be more complex to obtain from within LaTeX or by using the LaTeX generated coordinates.

With the process of obtaining the ground-truth concluded and with it the developing of the software system complete, the next step of was to validate the generated data and analyze it.

4.2 Data Validation

Bearing in mind that the ground-truth of the data set presented in this paper was generated automatically from a collection of scientific papers, there was no previous information about the position of MEs in these documents. In light of this fact, data validation had to be done by visually checking that the bounding boxes of the MEs present were in the correct position and of the right dimension.

For this reason, the validation of the data generated from this data set was one of the most time consuming steps.

It's important to say that since the ground-truth was generated automatically and given the high variability of the methods of typesetting MEs, some cases where the data extraction process would fail were expected.

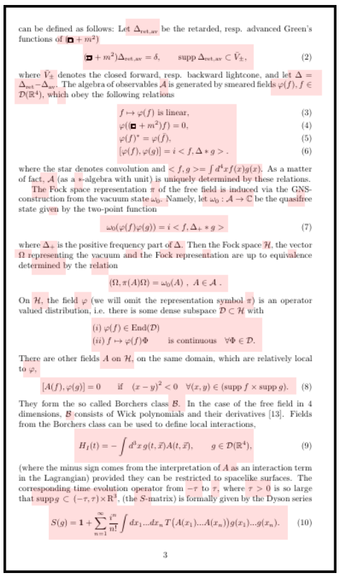
One of the first things we had to ensure, was that we did not modify the structure of the documents by inserting LaTeX commands in the definition of MEs. For this reason, the correctness of both the regular expressions and the LaTeX macros inserted had to be verified. Any undesired added space could shift elements of the MEs and invalidate the ground-truth of the document in question. Verifying if the layout of the original pdf format of the documents and their corresponding modified version where the expressions were highlighted was done by comparing both pdfs with the application *diffpdf* [2].

The *diffpdf* application allows comparison of pdf documents by appearance giving the option to use an XOR function to calculate and highlight the difference between the documents. Unfortunately this application could not be used to automatically scan the whole data set as it provided a boolean output for an exact match, and because it sometimes had pixel deviations when loading the documents which would invalidate the result. However, it was of great use for visually checking the difference between the documents because the pixel errors when loading the files were minor and did not affect the visual comparison. Figure 4.7 shows the comparison between the original and the highlighted version of page three of paper 0001129 shown in Figure 3.1. The difference between these two versions is shown by highlighting the pixels that do not match 4.7a, or by applying an XOR to the pixels of these pages 4.7b.

In Figure 4.7 it can be seen that the differences between the documents come from the colored pixels that form the bounding boxes of the MEs and some small pixel errors when loading the files.

After all the LaTeX macros and regex were slowly introduced and tested for errors, we started visually verifying the 957 papers, that did not rename the de-

[2] http://manpages.ubuntu.com/manpages/trusty/man1/diffpdf.1.html



can be defined as follows: Let $\Delta_{\text{ret,av}}$ be the retarded, resp. advanced Green's functions of $(\blacksquare + m^2)$

$$(\blacksquare + m^2)\Delta_{\text{ret,av}} = \delta, \qquad \text{supp}\,\Delta_{\text{ret,av}} \subset \bar{V}_{\pm}, \tag{2}$$

where $\bar{V}_{\pm}$ denotes the closed forward, resp. backward lightcone, and let $\Delta = \Delta_{\text{ret}} - \Delta_{\text{av}}$. The algebra of observables $\mathcal{A}$ is generated by smeared fields $\varphi(f), f \in \mathcal{D}(\mathbb{R}^4)$, which obey the following relations

$$f \mapsto \varphi(f) \text{ is linear,} \tag{3}$$
$$\varphi((\blacksquare + m^2)f) = 0, \tag{4}$$
$$\varphi(f)^* = \varphi(\bar{f}), \tag{5}$$
$$[\varphi(f), \varphi(g)] = i < f, \Delta * g > . \tag{6}$$

where the star denotes convolution and $< f, g >= \int d^4x f(x)g(x)$. As a matter of fact, $\mathcal{A}$ (as a $*$-algebra with unit) is uniquely determined by these relations.

The Fock space representation π of the free field is induced via the GNS-construction from the vacuum state ω_0. Namely, let $\omega_0 : \mathcal{A} \to \mathbb{C}$ be the quasifree state given by the two-point function

$$\omega_0(\varphi(f)\varphi(g)) = i < f, \Delta_+ * g > \tag{7}$$

where Δ_+ is the positive frequency part of Δ. Then the Fock space $\mathcal{H}$, the vector Ω representing the vacuum and the Fock representation are up to equivalence determined by the relation

$$(\Omega, \pi(A)\Omega) = \omega_0(A)\ , \ A \in \mathcal{A}\ .$$

On $\mathcal{H}$, the field φ (we will omit the representation symbol π) is an operator valued distribution, i.e. there is some dense subspace $\mathcal{D} \subset \mathcal{H}$ with

$$(i)\ \varphi(f) \in \text{End}(\mathcal{D})$$
$$(ii)\ f \mapsto \varphi(f)\Phi \qquad \text{is continuous} \quad \forall \Phi \in \mathcal{D}.$$

There are other fields A on $\mathcal{H}$, on the same domain, which are relatively local to φ,

$$[A(f), \varphi(g)] = 0 \qquad \text{if} \quad (x-y)^2 < 0 \quad \forall (x,y) \in (\text{supp}\, f \times \text{supp}\, g). \tag{8}$$

They form the so called Borchers class $\mathcal{B}$. In the case of the free field in 4 dimensions, $\mathcal{B}$ consists of Wick polynomials and their derivatives [13]. Fields from the Borchers class can be used to define local interactions,

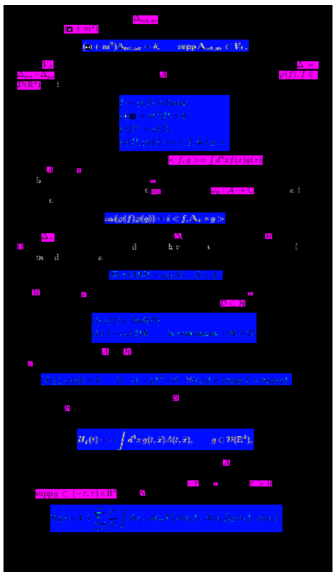
$$H_I(t) = -\int d^3x\, g(t, \vec{x})A(t, \vec{x}), \qquad g \in \mathcal{D}(\mathbb{R}^4), \tag{9}$$

(where the minus sign comes from the interpretation of A as an interaction term in the Lagrangian) provided they can be restricted to spacelike surfaces. The corresponding time evolution operator from $-\tau$ to τ, where $\tau > 0$ is so large that $\text{supp}\, g \subset (-\tau, \tau) \times \mathrm{R}^3$, (the S-matrix) is formally given by the Dyson series

$$S(g) = \mathbf{1} + \sum_{n=1}^{\infty} \frac{i^n}{n!} \int dx_1 ... dx_n\, T\big(A(x_1)...A(x_n)\big) g(x_1)...g(x_n). \tag{10}$$

3

(a) Highlighted difference.



(b) XOR difference.

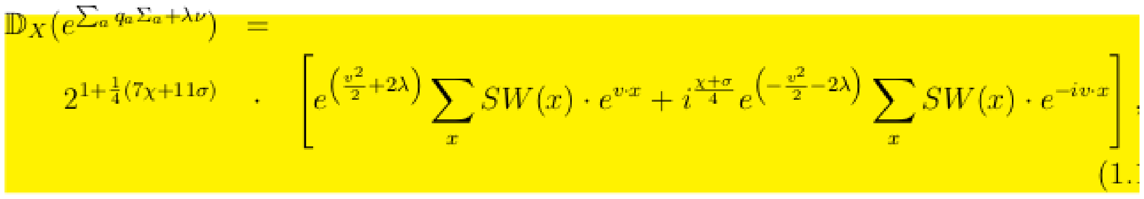
Figure 4.7: output generated with `diffpdf` when comparing the original and the highlighted version of page three of paper 0001129 [1].

limiters of the mathematical environments, in order to compile a collection of 200 documents.

When validating the data set, we encountered two types of errors, one that could be more easily corrected than the other. These errors were related to the use of regular expressions for automating the insertion of LaTeX macros.

Regular expressions have limitations in the sense that they tend to be too rigid and for this reason, given the high variability in writing styles, there were cases where macros were not inserted because of discrepancies in the pattern matching process. This type of errors could be fixed by creating new regex to take into account the special cases that we gradually encountered or by increasing the complexity of the regex already created in order to make them more flexible. The latter approach could be more dangerous because changes in the definition of regex could introduce unwanted behaviour in expressions that previously were correct. Once changes like this were made, the validation process had to be taken from the start in order to guarantee that documents that were previously validated did not suffer modifications.

The second type of errors we encountered were caused by macros that the authors had created in order to typeset subexpressions that were repetitive and large. If these subexpressions contained sums, products integrals or other elements with superscript or subscript symbols, the regular expressions would not be able to detect them and could cause inaccuracies in the highlighting of MEs. Figure 4.8 captures such a case.



$$\mathbb{D}_X(e^{\sum_a q_a \Sigma_a + \lambda\nu}) = 2^{1+\frac{1}{4}(7\chi+11\sigma)} \cdot \left[e^{\left(\frac{v^2}{2}+2\lambda\right)} \sum_x SW(x) \cdot e^{v \cdot x} + i^{\frac{\chi+\sigma}{4}} e^{\left(-\frac{v^2}{2}-2\lambda\right)} \sum_x SW(x) \cdot e^{-iv \cdot x}\right] \quad (1.$$

Figure 4.8: example of an error of highlighting a ME.

Considering that an error like the one presented in Figure 4.8 could invalidate the whole document and render the ground-truth unusable, we separated the highlighting phase from the extraction of the ground-truth. This modular architecture would allow for manual correction of errors in highlighting before launching the extraction of information from the data set.

Out of the approximately 1000 documents we set apart for creating the data set, we manually chose 200 documents that were visually validated and we proceeded with extracting the ground-truth. Table 4.1 highlights the characteristics of the resulting data set.

4.3 Complementary data set

Once we created the data set of MEs, we decided to expand the scope of our project and also create a data set of images of all the symbols and characters contained in these MEs. The approach we used was to process and tokenize the LaTeX definition of each ME in order to extract the symbols and characters that describe the expression.

We created a `lua` script, adapted from the `LuaTex` manual [15], that would be executed when LaTeX launches its output routine. This output routine of LaTeX consist of gradually making boxes of each element present, then combining these

Table 4.1: statistics about the data set.

Total no. of documents	200
Total no. of pages	2 460
No. of displayed MEs	16 598
No. of inline MEs	77 631
Average no. of pages per document	12.3
Average no. of displayed MEs per document	83.0
Average no. of inline MEs per document	388.2

boxes to create a line box, which combined with other lines, creates a paragraph box, and so on, until LaTeX obtains a container for the whole page by aligning and stacking smaller boxes. Each ME would be converted to a linked list of tokens, where each token would represent a glyph of the character/symbol to be rendered or the allocated space between individual characters/symbols and between words. Of these tokens we were only interested in the glyphs, where each symbol and character would have a different glyph, depending on the font used to typeset the given element. For transforming the written text into glyphs, LaTeX assigns to each character and symbol an entry to a font table containing all the glyphs defined for that font. Since we wanted to create individual LaTeX files of these glyphs which would later be compiled and converted to images, we saved both the table entry and the font name to be able to reproduce the exact same character or symbol.

After obtaining the LaTeX files, we analyzed their corresponding pdf representations and we noticed that there were many similar glyphs representing the same character or symbol, and that their font name or index was not the same. This posed a problem, because we could not use this information to be able to group these representations together and assign them the same class label. Since all the glyphs representing the same symbol or character had the same unicode, the solution was to map the corresponding font table indexes to the unicode code point of the glyphs to be able to group them together. The LaTeX distributions contain `.htf` files of different fonts with such mappings for nearly all the glyphs used. These files are under the *LaTeX project public license*, found at [16] and since they can be found in every LaTeX distribution, we will not provide them. These `.htf` files can also be generated as explained in [17].

By processing the expressions found in the 200 documents presented in this paper, we obtained 539.509 glyphs, representing a total number of 268 different symbols and characters. These symbols and characters follow an exponential distribution similar to Zipf's Law. In this case the frequency of the symbol `(` made up for $6, 71\%$ of the total number of glyphs, closely followed by the symbol `)` with 6.62% (as expected). The next most frequent character was the digit 1 with a percentage of 4.90%.

Once we were able to group up similar characters and symbols and correctly label them, we converted their pdf representations into images which we later centered, scaled and padded to a 28×28 pixels dimension. The exact format of this data set will be presented in chapter 5.

4.4 Data Analysis

In this section we will present a clustering experiment applied to the edit distance between pairs of MEs. This experiment is useful for understanding the frequency distribution of MEs, which could prove interesting for developing a search engine with tolerance where STEM documents are indexed by MEs.

The experiment had two parts. The first part involved identifying all the unique MEs in order to calculate the editing distance between every pair of MEs. The second part consisted of applying a clustering algorithm to the editing distances previously calculated with the purpose of grouping up MEs based on their similarity.

After obtaining the ground-truth, we analyzed the definitions of the MEs detected in the data set in order to calculate their reoccurrance frequency. Therefore, it was necessary to group up MEs based on their definition. However, considering that LaTeX is quite flexible and that it provides the user with many alternatives when it comes to typesetting MEs, an exact matching of the LaTeX transcript of MEs would not correctly identify different definitions with the same representation. We solved this problem by transforming each ME into its corresponding unicode symbol sequence.

When creating the complementary data set presented in section 4.3, we noticed that many LaTeX commands and symbols ended up having the same unicode code points, therefore we transformed each expression into its corresponding unicode symbol sequence in order to maximize the matching between different definitions of the same representation. We also had to take into consideration that the variables used to define a mathematical expression could vary from author to author. For this reason, after a preliminary matching, we computed the Levenshtein distance for every pair of expressions giving less weight to substitution. In Figure 4.9 we highlighted the substitution weight that we modified for

$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j) + 1 \\ \operatorname{lev}_{a,b}(i,j-1) + 1 \\ \operatorname{lev}_{a,b}(i-1,j-1) + \boxed{1_{(a_i \neq b_j)}} \end{cases} & \text{otherwise.} \end{cases}$$

Figure 4.9: Levenshtein distance with standard weights (source Wikipedia).

this experiment. By giving this weight a value of 0.75 instead of the standard 1 for deletion and/or insertion of mathematical symbols/characters, pairs of expressions in which one variable was replaced were more likely to be considered similar. As a result, an example such as $a + b + c = d$ is now more likely to be paired with the expression $a + b + c = e$.

Figure 4.10 shows the occurrence frequency of the 24 766 MEs resulting after the initial exact matching. This frequency distribution will be used as a baseline for interpreting the occurrence frequency of the MEs given an edit distance tolerance.

Considering that the degree of similarity between two expressions was highly dependant on the length of these expressions, we calculated the post-normalized edit distance by dividing the edit distance by the length of the longest expression

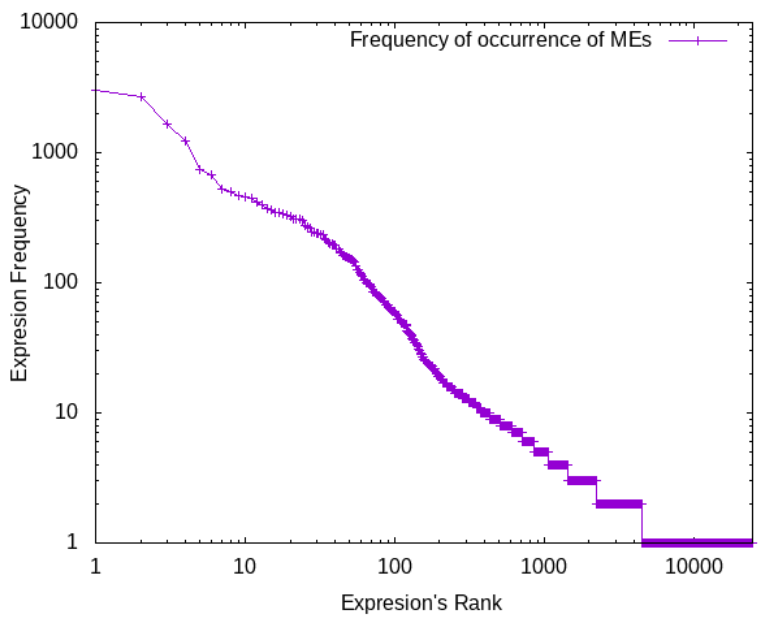


Figure 4.10: frequency of occurrence of the MEs after the initial matching.

of the two. Even though the post-normalized edit distance violates the triangular inequality and is sub-optimal, it has a computational cost of $O(nm)$ where n and m are the lengths of the two expressions. This computational complexity makes the post-normalized edit distance more efficient to calculate than the algorithm presented in [18] which as stated has "a linear increase in computational complexity with respect to the classical unnormalized edit distance procedure". Taking into account that there are approximately 25.000 different expressions after the initial matching, applying the optimal algorithm for calculating the normalized edit distance between every pair of mathematical expressions was not feasible.

In order to calculate the occurrence frequency of the expressions in the data set with a certain tolerance, we computed the post-normalized edit distance between every pair of MEs and stored it into a matrix. Given that such matrix was symmetrical, we only computed the upper triangular part of the matrix in order to avoid repetitive calculations. Once the pairwise distance between all MEs had been calculated with a substitution weight of 0.75 and a deletion and insertion weight of 1, we applied a clustering algorithm for grouping up similar MEs. We chose a substitution weight of 0.75 in order to balance the use of substitution when calculating the distance between two expressions. A weight lower than 0.75 would lead to incorrect results for pairs of expressions with the same length but different semantics, while a weight higher than 0.75 would not be sufficiently tolerant for pairing expressions with the same semantic but different variables.

Given that we could not know the number of clusters beforehand, we choose the DBSCAN clustering algorithm [19] that would calculate these clusters based on the density of the data, connecting MEs within a distance threshold of 20%. There were a total of 20 049 clusters obtained from 24 766 MEs. Figure 4.11 shows the frequency of the clusters resulting after grouping up similar MEs (green) against the frequency of MEs obtained in the initial matching process (magenta).

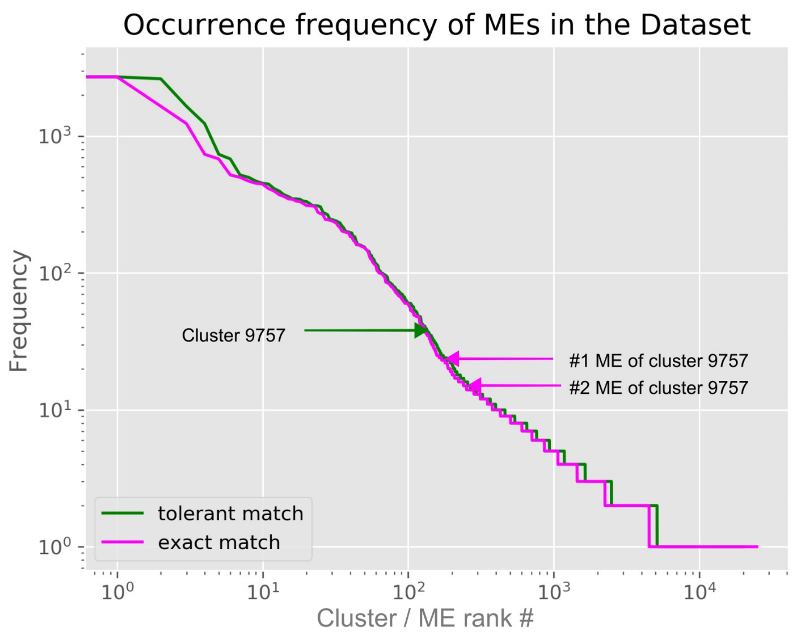
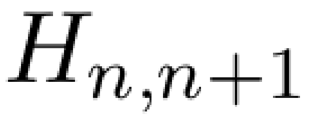


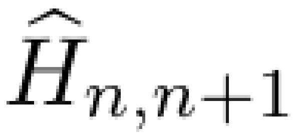
Figure 4.11: occurrence frequency of MEs and their corresponding clusters.

In order to demonstrate how the frequency of the clusters was calculated, we chose the cluster number 9757 that is made up of two MEs that are highlighted in Figure 4.11. The representation of these two MEs, with an occurrence frequency of 25 and 14 respectively, can be seen in Figure 4.12.

It is clear that the normalized edit distance between the MEs shown in Figure 4.12 is less than the 20% threshold we established for the clustering algorithm, with a difference of only one symbol. Grouping up MEs, such as the ones shown above, into clusters resulted in the redistribution of the occurrence frequency that can be seen in Figure 4.11.



(a) #1 ME of cluster 9757.



(b) #2 ME of cluster 9757.

Figure 4.12: MEs of cluster 9757.

Even though for this project, the clustering algorithm had been only applied to a Levenshtein distance calculated with a weight of 0.75, 1, 1 for substitution, insertion and deletion, respectively, the results of the experiment shown in Figure 4.11 illustrates the usefulness of enriching the ground-truth with the LaTeX transcript of MEs.

Given that the clustering algorithm returns the cluster label assigned to each ME, applying a clustering algorithm to the pairwise distance between the definitions of MEs can be of interest when implementing a search engine with a certain degree of tolerance.

CHAPTER 5

Structure of IBEM data set

The data set is distributed between three folders. The first folder consists of images obtained by breaking down pdfs of the original documents into images of each page. For every image, we provide a text file containing the coordinates and type of each one of the MEs detected in the corresponding page. Figure 4.6 gives an example of such data.

The second folder consists of a collection of sub-folders, where each sub-folder contains a scientific paper in LaTeX and pdf formats from which data was extracted. The name of each sub-folder starts with two digits indicating the publishing year of the paper, followed by five digits indicating the number assigned to the paper. Along with the original document we also provide the LaTeX file created by highlighting MEs, their pdf representation and a folder named *Indexing_results* where we include the output generated during each step of the process.

The third folder contains images of each symbol and character found in the definition of the MEs, a file containing a list with all the classes assigned to each image and a file with the values of the pixels in these images.

To facilitate the traceability and replicability of the methods described in this paper, we have not changed the names of the files extracted from the KDD data set.

5.1 Text Data

- A text file corresponding to each processed page is stored in the `Dataset` folder. The text file contains relative coordinates for every ME, stored as a *numpy* matrix as shown in Figure 4.6.
- In the *Indexing_results* folder corresponding to each paper:
  1. A text file with LaTeX coordinates for every inline ME detected.
  2. A text file with LaTeX coordinates for every displayed ME detected.
  3. A text file containing the definition of every inline ME detected.
  4. A text file containing the definition of every displayed ME detected.
  5. A text file containing the absolute page number of every inline ME detected.
  6. A text file containing the absolute page number of every displayed ME detected.
  7. A Json file with the coordinates, type, page number and LaTeX definition of every ME.
- A text file containing the class labels of each glyph contained in the definition of the MEs.

5.2 Image Data

The images described here are obtained with a density of 300dpi, a dimension of 1477×2048 pixels for the definition of the MEs and a dimension of 28×28 for the glyph representations of symbols and characters.

- An image file corresponding to each page of the original paper is stored in the *data set* folder.
- In the *Indexing_results* folder corresponding to each paper:
  1. An image file for every page of the highlighted document as shown in the second image of Figure 3.1.

2. An image file for every page of the original document consisting of a negative containing only the detected MEs, as shown in the third image of Figure 3.1.
3. An image file per ME detected, representing the renderization of the expression.

- An image file corresponding to each symbol and character. The frequency and class of these symbols and characters are codded into the image's name.

CHAPTER 6

Experimental Results

This chapter introduces an experiment based on this data set as a baseline. Several types of experiments can be carried out with this data set, given that it has GT at different levels: ME detection and extraction, ME recognition, ME searching, etc. This experiment targets ME recognition, a step that takes place once the definition of MEs has been detected and extracted. By performing a symbol classification experiment we would lay the foundation for interpreting MEs while illustrating the possible problems when developing a ME recognition system. Our goal was to perform an experiment that could be easily replicated. The motivation behind the experiment is based on our desire to illustrate one of the possible uses of including the LaTeX transcript of MEs into the ground-truth. Since the GT of the data set already facilitates ME detection and extraction, the next challenge would have been ME recognition. To tackle this challenge, symbol classification was necessary. Therefore, the goal of this experiment was to provide a starting point for solving the ME recognition challenge. This symbol classification experiment was conducted with current technology based on Convolutional Neural Networks (CNN).

6.1 Classification Model

Mathematical symbol classification experiments were performed with a CNN with a usual architecture. This model has the ability to learn key spatial features of an image and understand the sophistication of the image, that is, differentiate between the symbols given the inherent variability of each class as a result of changes in font representations of these symbols.

We built a sequential model with 3 convolutional layers of 32, 64 and 128 filters respectively. For all these layers we defined a convolutional window of dimension 3×3 with a stride of 1×1 along the x and y axis of the input image. We also activated zero padding to allow the kernel to process the entire surface of the images, given that the size of each such image is 28×28 pixels. We chose `ReLU` as the activation function to be applied after the convolutional process of these layers. In order to reduce the dimensionality of the data and extract dominant features which are rotational and positional invariant, following each convolutional layer we placed a Max Pooling Layer, given that they usually exhibit a better result [20] than other pooling layers. To reduce over fitting we introduced a dropout regularization of 25% after each pooling layer. As the last layer we added a single fully connected layer with 268 neurons with a `softmax` activation classifier. Figure 6.1 captures the architecture of the CNN model.

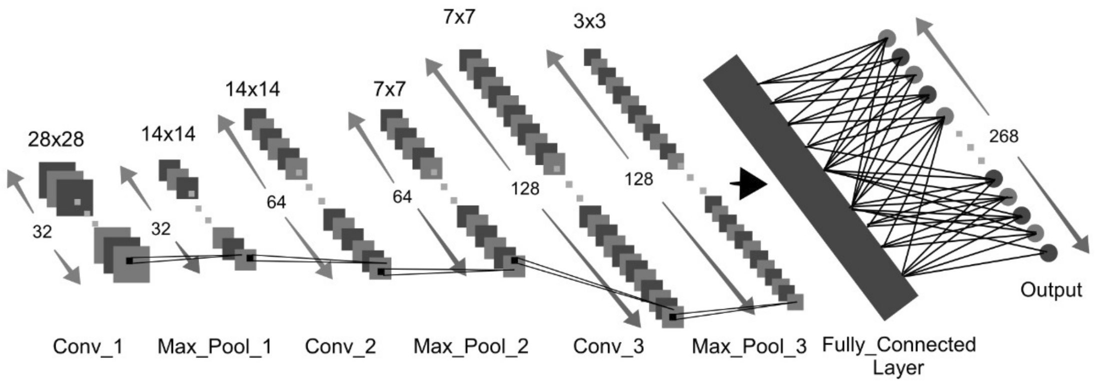


Figure 6.1: CNN architecture.

We chose *accuracy* as a metric and a *categorical cross-entropy* as a loss function for optimizing the predictive model. After choosing a batch size of 100 images and 10 epochs we started training the network.

The training process of the CNN, that can be observed in Figure 6.2, indicates a rapid convergence of the model in under 10 epochs. While the learning rate is quite high, we obtained a validation accuracy of 0.9991 and a validation loss of 0.0544 with little to no over-fitting.

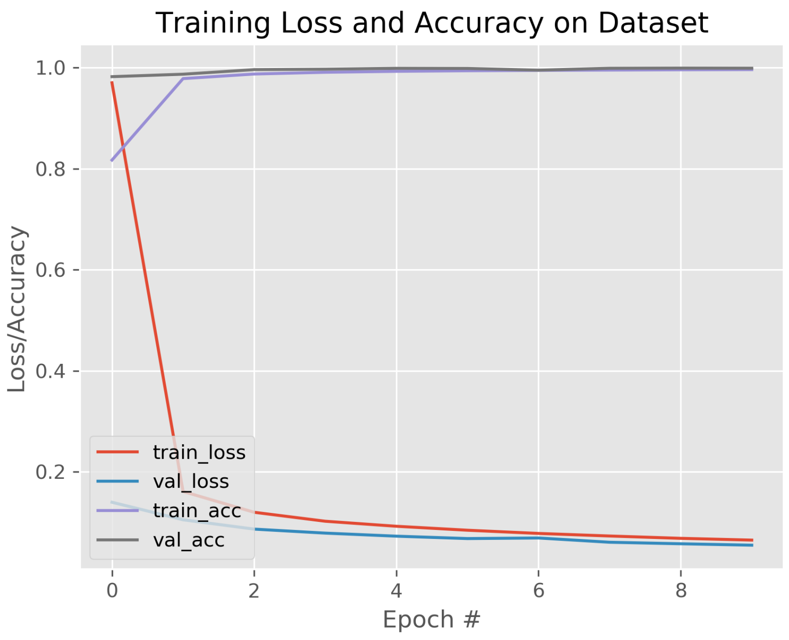


Figure 6.2: CNN training phase.

6.2 Data Statistics

As mentioned before, there are a total number of 539 509 images in this data set, one per glyph encountered in the definition of the MEs of the 200 documents. Since we decided to carry out the classification experiment as simple as possible, the glyphs were grouped in the same class if they represented the same symbols, that is, different fonts of the same symbols were grouped in the same class. This is a clear simplification because sometimes MEs have the same symbol with a different meaning. In this way, the number of classes was reduced to 268.

We partitioned the data assigning 80% for training and 20% for testing. Out of those 80% images, we chose a 20% subset for validation. Table 6.1 shows the main statistics.

6.3 Results and Discussion

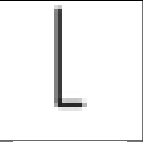
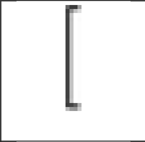
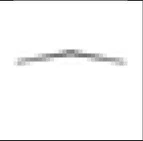
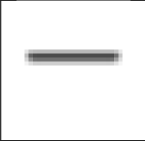
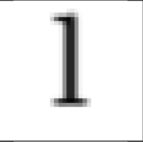
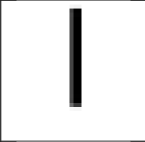
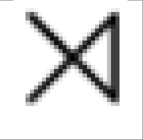
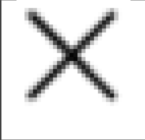
The final result on the test set after training the CNN model was 0.07% error rate. Note that this is a very good results, but we have to take into account the the amount of training data was really large.

Table 6.1: partitions of the data set.

No. of images in training set	345 285
No. of images in validation set	86 322
No. of images in test set	107 902

Out of the 107 902 images in the test set we had 80 cases of misclassification. Out of these 80 errors, 18 were a result of assigning a different class to the same character when labeling the data set. This happened due to the fact that characters had different font representations and their font table entries were missing from the mapping files we used to translate the font glyphs into their corresponding unicodes. By modifying and correcting these font files, we could have improved the accuracy by 0.01%. Of the remaining 62 misclassifications, we provide some examples to illustrate where the classifier learned wrong features that led to errors in the prediction of the test classes. Table 6.2 shows some of these misclassification examples.

Table 6.2: test symbol vs. predicted symbol

These are some of the most representatives examples of errors of prediction that we encountered by analyzing the prediction output generated by the neural network. The first example represents many of the variants of left/right floor and left/right ceil symbols being incorrectly classified as open or closed brackets.

There were a total of 15 such cases. There were 6 cases of the second example. These misclasifications are a result of the rotational and translational invariance characteristic of convolutional networks. The third example could be considered very similar to the first one, where a small feature could significantly change the meaning of the symbol. There were 6 test samples that fit this case. It is important to know that the occurrence frequency of this exact glyph representing the letter `l` was 737 and of its class 4 673, while the frequency of the symbol | was 472, therefore this was not a sign of over-fitting of the model.

CHAPTER 7

Future work

This chapter will explore potential improvements that could be brought to the process of obtaining the ground truth, as well as suggesting additional information for enriching the ground-truth already obtained.

One of the key areas that could be improved relates to the module for detecting and highlighting MEs. Since this module precedes the ground-truth extracting section, any improvements brought to it will directly improve the accuracy of the ground-truth. There are two ways to improve the MEs detecting and highlighting method used in this paper. The first relates to expanding the number of environments for which detection and highlighting can be done with the methods employed in this project, while the second relates to detection of MEs that have been defined using sub-macros.

One of the drawbacks of employing regular expressions for detecting MEs, is the lack of adaptability to new ways of typesetting ME. These regular expressions need to be adapted and maintained in order to provide full support to changes in the definition of mathematical environments. This project provides regexes for the most common environments, therefore new sets of regular expressions need to be developed in order to provide functionality for environments we do not cover.

Related to the highlighting process, we found that there are authors that define LaTeX macros to facilitate typesetting of repetitive sub-expressions. As mentioned in Chapter 4 these macros could cause inaccuracies in the highlighting of MEs. These imprecisions occur when sub-expressions, that have been defined with macros, contain complex elements with superscript or subscript symbols that could be rendered at a different height than the rest of the symbols defining the ME. This situation would cause the regular expressions in charge of detecting such sub-expressions to fail, which would result in not inserting coordinate points to include these elements when calculating the position of the bounding box of the ME. Currently, situations such as this are detected by measuring the bounding boxes of MEs independently of the position on the page. If the distance between the upper left corner and lower right corner coordinate is not equal to the dimensions we measured, there must be an element not taken into account when calculating the coordinates. Because we could not know the position of such an element, we split the height difference between the upper and lower part of the bounding box, reducing the inaccuracies of the highlighting process. This process could be improved by parsing the documents beforehand and replacing all instances of the LaTeX macros created by the authors with their definition. This would allow the detection of sub-expressions and would improve the accuracy of the highlighting by adjusting the bounding boxes more tightly around MEs.

Another improvement that should be explored is the automation of the validation process. Currently this process is done manually, thus if the aim is to create a data set with thousands of documents, the validation phase would take a significant amount of time and people, not to mention possible errors that could be committed due to fatigue.

In addition to the ground-truth already presented in this paper, we consider that the definitions of MEs could be further analyzed, which would provide interesting information for enriching the data set. Besides extracting the sequence of unicode code points, representing the glyphs of characters and symbols defining the MEs, it is also interesting to extract information such as the location of these symbols and characters in relation to the ME, and also if they are rendered as a superscript or subscript element. Information such as this could prove invaluable when studying the relationship between the symbols of a ME.

Considering that the order of the elements of MEs is important, which gives these expressions semantic meaning, understanding the relationship between these elements could allow us to transform MEs through mathematical properties. Having this information would allow for building a ME search engine more robust, providing more flexibility and tolerance for matching queries.

CHAPTER 8
Conclusions

As stated, the objective of this paper was to generate a labeled data set derived from digital STEM documents containing mathematical expressions. The data set had to include information about the location and the content of said mathematical expressions. Several specific objectives have been targeted, amongst which: developing a software system for GT extraction, building a secondary data set of symbol and character images and conducting an analysis of ME occurrence frequency.

In reaching these objectives, several challenges occurred, such as the diversity of methods for typesetting mathematical expressions in LaTeX, the availability of large volumes of STEM documents, as well as potential copyright issues derived from processing them. We consider these challenges solved within the boundaries of the current project. Other challenges, such as adjusting the identification software to include more environments with which MEs can be written in LaTeX or adjusting it to correctly deal with situation where repetitive sub-expressions are defined using macros remain to be solved.

Out of the approximately 1000 documents that have been processed for this project, we've met the 200 document threshold set as the primary objective. As a result, the identification software has the desired accuracy of 20%. It should be stated, that since the results of the highlighting process have to be checked manually, which is time consuming and effort intensive, the real accuracy of the software could be much higher than the targeted 20%. In fact, based on further investigation we consider that the accuracy of the software is greater than 60%, but for the purposes of this project, the lower threshold was considered sufficient.

In relation to the secondary data set of symbol and character images extracted from the MEs, we have obtained a set of 539 509 images, corresponding to 268 unique symbols. This secondary data set allowed us to calculate the prior probability of each symbol.

The ME analysis experiment has resulted in valuable output that can be used as a starting point in building a tolerant search engine of MEs within large collections of digital documents.

Based on this information, we consider that the objectives of the project have been successfully met.

On a personal note, the project represented a means of exploring data set creation realities. On this level, my personal objective has also been met and I believe I have gained a much better grasp of data set creation and its challenges and opportunities.

This project has been developed as part of a larger project, IBEM [21]. An article based on the project presented in this paper has been submitted for publication in the ICPR2020 conference [22]. This article has passed the first round of a two-round review system, and is currently under the process of revision.

We have also put forth the idea of organizing a ME detection competition starting from a subset of the corpus presented in this paper. So far this idea has been presented to ICPR unsuccessfully, and it is scheduled to be submitted to ICDAR conference taking place in 2021. The results of the second submission are currently not available.

Once the IBEM project is finalized, we will commit to uploading the complete corpus resulted from this project to the ZENODO platform (and/or GitHub).

Acknowledgment

This work has been partially supported by the Ministerio de Ciencia y Tecnología under the grant TIN2017-91452-EXP (IBEM) and by the Generalitat Valenciana under the grant PROMETEO/2019/121 (DeepPattern).

Bibliography

[1] M. Dütsch and K. Fredenhagen, "Algebraic quantum field theory, perturbation theory, and the loop expansion," *Communications in Mathematical Physics*, vol. 219, no. 1, pp. 1–26, 2001.

[2] M. Mahdavi, R. Zanibbi, H. Mouchère, C. Viard-Gaudin, and U. Garain, "ICDAR 2019 CROHME + TFD: Competition on recognition of handwritten mathematical expressions and typeset formula detection," in *International Conference on Document Analysis and Recognition*, 2019.

[3] Y. Deng, A. Kanervisto, and A. M. Rush, "What you get is what you see: A visual markup decompiler," *ArXiv*, vol. abs/1609.04938, 2016.

[4] W. Ohyama, M. Suzuki, and S. Uchida, "Detecting mathematical expressions in scientific document images using a u-net trained on a diverse dataset," *IEEE Access*, vol. 7, pp. 144 030–144 042, 2019.

[5] F. Álvaro and J. A. Sánchez, "Comparing several techniques for offline recognition of printed mathematical symbols," in *2010 20th International Conference on Pattern Recognition*, 2010, pp. 1953–1956.

[6] I. Phillips, "Methodologies for using UW databases for OCR and image understanding systems," in *Proc. SPIE, Document Recognition V*, vol. 3305, 1998, pp. 112–127.

[7] F. Shafait, J. Beusekom, D. Keysers, and T. Breuel, "Page frame detection for marginal noise removal from scanned documents," vol. 4522, 06 2007, pp. 651–660.

[8] M. Suzuki, S. Uchida, and A. Nomura, “A ground-truthed mathematical character and symbol image database,” in *Proc. 8th International Conference on Document Analysis and Recognition (ICDAR’05)*, 2005, pp. 675–679.

[9] J. Gehrke, P. Ginsparg, and J. Kleinberg, “Overview of the 2003 kdd cup,” *SIGKDD Explor. Newsl.*, no. 2, p. 149–151, Dec. 2003.

[10] American Mathematical Society, *Amsmath Package*, 1999. [Online]. Available: https://osl.ugr.es/CTAN/macros/latex/required/amsmath/amsldoc.pdf

[11] L. Madsen, “Avoid eqnarray!” *PracTeX*, no. 4, 2006.

[12] K. Pizzini, P. Bonzini, J. Meyering, and A. Gordon, *GNUsed, a stream editor*. [Online]. Available: https://www.gnu.org/software/sed/manual/sed.pdf

[13] Heiko Oberdiek, *The zref package*. [Online]. Available: https://osl.ugr.es/CTAN/macros/latex/contrib/zref/zref.pdf

[14] T. Tantau, *The TikZ and PGF Packages*. [Online]. Available: http://sourceforge.net/projects/pgf/

[15] *LuaTEXReferenceManual*. [Online]. Available: http://www.pragma-ade.com/general/manuals/luatex.pdf

[16] “Latex project public license,” [Last Access 13-April-2020]. [Online]. Available: https://www.latex-project.org/lppl/

[17] “Font mapping generation - htfgen,” [Last Access 13-April-2020]. [Online]. Available: https://github.com/michal-h21/htfgen

[18] A. Marzal and E. Vidal, “Computation of normalized edit distance and applications,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 9, pp. 926–932, 1993.

[19] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *KDD*, 1996.

[20] V. Suárez-Paniagua and I. Segura-Bedmar, “Evaluation of pooling operations in convolutional architectures for drug-drug interaction extraction,” *BMC Bioinformatics*, vol. 19, no. 209, 2018.

[21] E. Vidal, F. Casacuberta, J. A. Sanchez, and J. M. Benedí, "IBEM: Indexing and search of mathematical expressions on a large scale in massive corpus of printed documents," to appear.

[22] D. Anitei, J. A. Sanchez, and J. M. Benedí, "A scalable printed document image dataset for indexing and searching mathematical expressions," submitted.

APPENDIX A

System Configuration

In this appendix we will cover the architecture of the software system presented in this paper by briefly presenting the components that form it. In order to be able to launch the program into execution and to be able to reproduce the results obtained in this project some software resources need to be available by installing or configuring them. In this chapter we will also explain how to execute individual functionality modules of the system if such a need arises.

A.1 Software Structure and Requirements

The software system built in this project consists of a series of scripts, written in shell and python code, that are linked together in order to provide the functionality described in this paper. This system is divided in two. One part is in charge of creating the data set of STEM documents and extracting the ground-truth from this documents (presented in chapter 4), while the second part is in charge of analyzing the sequence of symbols that form the MEs that were extracted from the previous data set, and creating a complementary data set of images of glyphs representing these symbols (presented in chapter 4.3). This secondary data set is then used in the training and testing of a symbol classifier presented in chapter 6.

Of the component in charge of highlighting MEs and extracting ground-truth the most important scripts are:

- *main_script.sh*, that as the name suggest is the main entry point of this component. The script is written in bash and offers access to the different modules that provides the functionality of the component. This script processes one document per execution,

- *lanza.sh*, bash script created for processing more than one document by launching the *main_script.sh* in a sequential manner,
- *find_coordinates.py*, python script for creating negatives like the one shown in Figure 3.1c and for extracting coordinates like the ones shown in Table 4.6,
- *createDB.sh*, bash script for creating individual LaTeX files for each one of the MEs extracted from the IBEM data set,
- *latex2png.sh*, bash script that as the name suggest, converts the LaTeX file of each ME into png of its corresponding graphical representation,
- *SED.txt*, SED script for inserting commands and calls to LaTeX macros for extracting the LaTeX transcripts of the MEs detected in the IBEM data set,
- *SED_cajas.txt*, SED script for inserting commands and calls to LaTeX macros for highlighting MEs,
- *useful_macros.sty*, library package with the definition of the LaTeX macros used for extracting and highlighting MEs,
- *symbols.lua*, LUA script for processing MEs and breaking them down into sequences of glyphs used for creating the data set of mathematical symbols presented in 4.3.

Of the second component in charge of analyzing the sequences of symbols that form the MEs, and creating a complementary data set of images of glyphs, the most important scripts are:

- *proc_symbols.py*, python script for creating the data set of mathematical symbols.
- *CNN.py*, python script for creating and training a CNN mathematical symbol classifier.

All the scripts presented above require that some software resources be present for a correct execution. It is important to know that this project has been developed under the Ubuntu Operating System, therefore all the following software resources that we detail were configured for this OS:

- *SED*, a stream editor for performing text transformations by means of regular expressions.
- *Python 3.7*, interpreted programming language.
- *OpenCV 3.2+*, an open source computer vision library used in this project for performing shape analysis on the highlighted MEs.
- *LaTeX*, a document preparation system with a descriptive markup language.
- *ImageMagick 6.9.10-23*, tool for creating, editing, composing, or converting bitmap images.
- *pdfinfo 0.86.1*, a Portable Document Format (PDF) document information extractor.
- *Keras 2.3.1*, a Python deep learning framework built on top of TensorFlow.
- *TensorFlow 2.1.0*, an end-to-end open source platform for machine learning.
- *CUDA Version: 10.2*, a parallel computing platform and application programming interface (API). Prerequisite for using TensorFlow.
- *cuDNN Version: 7.6*, a GPU-accelerated library of primitives for deep neural networks. Prerequisite for using TensorFlow.

A.2 Execution Modules and Prerequisites

The scripts presented in the anterior section form up the different modules that the project was divided into. As mentioned before, the main entry point for accessing these modules is through the bash script file *main_script.sh*. This script can receive several arguments that control which modules will be launched during the execution of the program, with a syntax like the following:

main_script.sh LATEX_SOURCE [OPTIONS]

where LATEX SOURCE is a mandatory argument containing the path to the LaTeX file to be processed, and the optional arguments are:

'-c': option for compiling the LaTeX source file, highlighting the bounding boxes of MEs (example shown in Figure 3.1b), and extracting the LaTeX transcript of these MEs.

Prerequisite: the main LaTeX file of the document to be compiled needs to be in an individual folder, separated from the other documents.

'-m': option for breaking down the PDF format, resulting from the compilation phase, into images of each page which are then processed in order to extract the coordinates of each bounding box.

Prerequisite: the LaTeX file has to have been previously compiled in order to generate the PDF format with the bounding boxes of MEs highlighted (option '-c').

'-o': option with the same functionality of option '-o', however the ground-truth is better structured (example shown in Figure 4.6). A negative of the images with just the definition of MEs is also generated with this option (example shown in Figure 3.1c).

Prerequisite: the LaTeX file has to have been previously compiled in order to generate the PDF format with the bounding boxes of MEs highlighted (option '-c').

'-l': option for creating individual LaTeX files and png representations of each MEs highlighted in the document.

Prerequisite: the LaTeX file has to have been previously compiled in order to extract the LaTeX transcript of MEs (option '-c').

'-s': option for generating statistics about the number of displayed and inline MEs detected, and the number of pages of the document. If the option is activated through launching the *lanza.sh* script, it also generates global statistics about the average number of inline and displayed MEs per document, and the total number of expressions and pages of the data set.

Prerequisite: the LaTeX file has to have been previously compiled in order to extract the LaTeX transcript of MEs (option '-c').

'- t': option for processing the LaTeX transcript of each ME, previously extracted, in order to generate sequences of glyphs of each symbol and character that composes the definition of the ME.

Prerequisite: the LaTeX file has to have been previously compiled in order to extract the LaTeX transcript of MEs (option '-c').

'- a': enables all of the above options.

If more than a document needs to be processed, then the recommended script to be used is *lanza.sh*. This script has a syntax similar to the one presented for *main_script.sh*:

lanza.sh DIR_SOURCE [OPTIONS]

where DIRECTORY SOURCE contains the path to a directory containing subfolders with individual LaTeX files to be processed, and the optional arguments are the same as the ones for the script *main_script.sh*.

While the creation of the IBEM data set and its associated ground-truth extracted from MEs can be obtained by launching *lanza.sh* or *main_script.sh* with the appropriate options, the creation of the complementary data set of mathematical symbols is obtained by launching the python script *proc_symbols.py*. This script requires a text file containing information about the font name and font table entry of every symbol and character found in the definition of MEs extracted from the IBEM data set. This file is generated by launching the processing of STEM documents with the option *-t*. The syntax of the script *proc_symbols.py* is as follows:

proc_symbols.py DST_PATH DIR_MAPPINGS

where DST_PATH is the directory path where the data set of symbols will be created, and DIR_MAPPINGS is the directory containing mapping files for converting glyphs to unicode code points.

Lastly, in order to train and create the CNN clasiffier we need to launch the python script *CNN.py* that accepts only one argument:

```
CNN.py data set_DIR
```

where data set_DIR is the directory containing the data set created with the script *proc_symbols.py*.

APPENDIX B

Regular Expressions

In this appendix, we will introduce some of the most important regular expressions used for inserting LaTeX macros and commands. These macros and commands, as explained before, had the function of calculating and highlighting the bounding box of MEs or the function of extracting the LaTeX transcript of MEs. In this appendix we will only present regular expressions related to the highlighting process given that the regexes developed for extracting the LaTeX transcripts are simplifications of the former.

This appendix might result harder to follow for persons not familiar with *SED's* syntax [1] or LaTeX typesetting [2]. All of the regular expressions presented in this appendix can be found in the *SED_cajas.txt* file.

First of all, we will introduce the regex used for importing the LaTeX package file containing the definition of the LaTeX macros used in this project:

```
/^%/! s/\\begin{document}.*/\\usepackage{useful_macros}\n&/g
```

This regex has two parts, where the first part conditions the second. The first part checks that the line to be processed does not start with the commentary symbol '%'. With this check, lines that have been commented out by the authors are ignored in order to make sure we only import the package file *useful_macros* once. In order to insert the loading of the package, we use *SED's* command substitute with the syntax *'s/regex/replacement/flags'*. Therefore, in the second part of

[1] https://www.gnu.org/software/sed/manual/sed.html

[2] https://en.wikibooks.org/wiki/A_Brief_Introduction_to_the_LaTeX_Typesetting_Environment/Typesetting:_The_Basics

the regex we search for the LaTeX command `\begin{document}` (seen in the regex part and denoted by `'&'` in the replacement part of the substitute command) and prepend the LaTeX command for loading package files. The flag used in this case and in the majority of the regular expressions shown in this appendix is the `'g'` flag that stands for global replacement of all the matches of the regex.

The two regular expressions we are going to present next, are the ones in charge of inserting the macros shown in yellow in Figure 4.4:

```
s/\\begin{eqnarray\*\?}/&\\coord{}\\boxAlignEqnarray{\\leftCoord{}/g
s/\\end{eqnarray\*\?}/}{0mm}{1}{1}{control}{theequation}{1}\\coordE
    {}&\%\%/g;
```

The first regex matches the beginning of the `eqnarray` environment and inserts the macro for the starting coordinate and the macro for highlighting and computing the dimension and location of the expression (`\boxAlignEqnarray`). The second regex matches the ending of the `eqnarray` environment and inserts the default arguments of the highlighting macro. The *"control"* and *"theequation"* arguments are used as marks for correctly identifying the rest of the arguments in order to be able to modify them as needed. The argument *"theequation"* is replaced with the definition of the ME as written by the author before inserting the calls to the macros we created. Given that we insert some of these macros before copying the definition of the expression, we created a regex for filtering out these macros after making the copy. The *"control"* mark is deleted once the arguments shown as [1][1] are overwritten with the number of left and right coordinates to be processed when computing the dimensions and location of the bounding box.

The following regular expressions are the ones that pass and clean the definition of MEs:

```
s/\(\\begin{eqnarray.\?}\)\([^%]*\)\(}{0mm}{..\?}{..\?}{control}{\)
    theequation\(}{1}\\coordE{}\\end{eqnarray\*\?}\)/\1\2\3\2\4/g;
:repeat1
        s/\({..\?}{control}{[^%]*\)\\rightCoord{}\([^%]*}\\coordE{}\\
            end{\)/\1\2/g;
    /{control}{[^%]*\\rightCoord{}/t repeat1
:repeat2
```

```
6        s/\({..\?}{control}{[^%]*\)\\leftCoord{}\([^%]*}\\coordE{}\\end{\)/\1\2/g;
7     /{control}{[^%]*\\leftCoord{}/t repeat2
8  :repeat3
9     s/\({control}{[^%]*\)\n*\s*\\p\?label\s*{[^}]*}\s*\n*\([^%]*}\\coordE{}\\end{\)/\1\2/g;
10    /{control}{[^%]*\\p\?label\s*{/t repeat3
```

There are four sets of regular expressions shown in the code above. The first one, numbered as 1, is the regex in charge of making a copy of the ME. This regex detects everything defined in the `eqnarray` environment and separates it into blocks of LaTeX code. Since we previously inserted the call to the `\boxAlignEqnarray` macro with the default corresponding arguments, we know that the ME is defined before the *"control"* mark. Therefore, we copy the LaTeXcode that precedes the control mark (shown in the regex as \2), overwriting the *"theequation"* argument (shown in the regex between the blocks \3 `and` \4). After copying the definition of the ME, we created regular expressions that filter this definition by looping over it as long as there still are `\rightCoord{}` macros (lines 2 to 4), and `\leftCoord{}` macros (lines 5 to 7). The third set of regular expressions (lines 8 to 10) has the task of eliminating the labels defined in the ME, since duplicating these labels would cause compiling errors.

There are similar regular expressions to the ones presented above, defined for all the mathematical environments that we presented in chapter 4.

Once the definition of the copy of the ME had been cleaned, we could count the number of left and right coordinate macros present in the mathematical environment. Given that stream editors like *SED* don't have arithmetic capabilities, we had to create as many regular expressions as we wanted to detect:

```
s/\(\(\\leftCoord{}[^%]*\)\{1\}\)\({[^}]*}\){..\?}\({..\?}{control}\)/\1\3{1}\4/g;
s/\(\(\\leftCoord{}[^%]*\)\{2\}\)\({[^}]*}\){..\?}\({..\?}{control}\)/\1\3{2}\4/g;
...
s/\(\(\\leftCoord{}[^%]*\)\{79\}\)\({[^}]*}\){..\?}\({..\?}{control}\)/\1\3{79}\4/g;
```

```
s/\(\(\\leftCoord{}[^%]*\)\{80\}\)\({[^}]*}\){..\?}\({..\?}{control}\)
    /\1\3{80}\4/g;
```

In the code shown above, it can be seen that each regular expression detects a different number of left coordinate macros, ranging from 1 to 80. A similar set of regular expressions has been created for detecting the right coordinate macros.

Considering that the number of left and right coordinate macros we can detect is limited by the number of regular expressions created, there can be the case of MEs where the number of coordinate macros inserted exceeds the number of coordinate macros detected. If such a case is given, there might be problems with computing the correct location of the bounding box since not all the coordinates will be processed. This problem would appear only if the ending coordinate is not an extreme point in the definition of the ME.

The rest of the regular expressions in charge of inserting coordinate macros between the symbols of some mathematical elements such as fractions, sums, products, integrals, etc. or before and after each newline symbol, can be found in the *SED_cajas.txt* file.