An Integrated Grammar-based Approach for Mathematical Expression Recognition

Francisco Álvaro, Joan-Andreu Sánchez, José-Miguel Benedí

Pattern Recognition and Human Language Technologies Research Center
Universitat Politècnica de València, 46022 Valencia Spain

Abstract

Automatic recognition of mathematical expressions is a challenging pattern recognition problem since there are many ambiguities at different levels. On the one hand, the recognition of the symbols of the mathematical expression. On the other hand, the detection of the two-dimensional structure that relates the symbols and represents the math expression. These problems are closely related since symbol recognition is influenced by the structure of the expression, while the structure strongly depends on the symbols that are recognized. For these reasons, we present an integrated approach that combines several stochastic sources of information and is able to globally determine the most likely expression. This way, symbol segmentation, symbol recognition and structural analysis are simultaneously optimized. In this paper we define the statistical framework of a model based on two-dimensional grammars and its associated parsing algorithm. Since the search space is too large, restrictions are introduced for making the search feasible. We have developed a system that implements this approach and we report results on the large public dataset of the CROHME international competition. This approach significantly outperforms other proposals and was awarded best system using only the training dataset of the competition.

Keywords: Mathematical expression recognition, probabilistic parsing, handwriting recognition

1. Introduction

Mathematical notation constitutes an essential source of information in many fields. Recognizing mathematical expressions is an important problem in scientific document recognition or the acceptance of mathematical expressions in human-based interfaces [1]. Lately, other applications like accessibility for disabled people and, especially, information retrieval are receiving more attention [2].

*Corresponding mail: falvaro@prhlt.upv.es

Email addresses: `falvaro@prhlt.upv.es` (Francisco Álvaro), `jandreu@prhlt.upv.es` (Joan-Andreu Sánchez), `jbenedi@prhlt.upv.es` (José-Miguel Benedí)

In this pattern recognition problem we usually distinguish between online and offline expressions. Offline formulas are represented as images and can be printed or handwritten. Meanwhile, an online expression is encoded as a sequence of points in space, and therefore includes time information. Handwritten expressions generally present more variability than printed expressions, while online data normally yield better results than offline expressions because they contain additional information [3]. This paper is focused on automatic recognition of online handwritten mathematical expressions.

The recognition of math notation is traditionally divided into three problems [1, 2]: symbol segmentation, symbol recognition and structural analysis. In the literature, there are two main approaches: sequential (decoupled) solutions and integrated solutions.

Sequential approaches tend to first look for the best segmentation of the input expression into math symbols. The analysis of the structure is then carried out on that best symbol segmentation [4]. This type of solution is not able to solve errors made during the first recognition stage if hard decisions are taken.

Integrated approaches set out to use the global information of the mathematical expression to obtain the final structure of the formula as a whole [5, 6]. The symbol segmentation and the symbol recognition is then obtained as a byproduct of the global optimization. These approaches seem more appropriate because obtaining the best symbol segmentation as a byproduct depends on the structure of the expression, and vice versa.

In this paper we present a novel integrated approach for the recognition of online handwritten mathematical expressions. This approach is inspired in the *holistic* approach used in *off-line* Handwritten Text Recognition (HTR) [7]. In the holistic HTR approach, the recognition process is modeled in several perception levels integrated in a unique model: optical models (Hidden Markov Models [8] or Recurrent Neural Networks [9]) are used for modeling characters in the low level; finite-state models are used for modeling words in the middle level [8]; and n-gram models are used for language modeling in the high level.

We follow an analogous approach in this paper with different perception levels integrated in a unique model. We describe a statistical framework based on two-dimensional grammars for the highest level since they constitute a natural and powerful model for dealing with structured problems. Mathematical symbols are used in the middle level that are modeled as sets of strokes, and strokes as primitives in the lowest level. It is important to remark that other input units smaller than strokes could be considered in the lowest level (like subsets of consecutive input points) but the principal ideas of the approach described in this paper would be the same.

We define the associated parsing algorithm that globally determines the most likely math expression based on several sources of information. The aspects that allow us to obtain the segmentation and recognition of symbols as a byproduct are explained in detail.

In the integrated approach, the search space becomes enormous and therefore we also present techniques based on spatial and geometric information for effectively reducing the search space. We impose restrictions based on the distance between strokes, and during the structure analysis we impose restrictions following the idea of hierarchical clustering algorithms. Finally we tackle the estimation of all the probability distributions.

The system that implements this proposal was awarded best system using the training set of the recent CROHME international competition [10], and has been released as open-source software. Here we report results on the large public dataset of this competition.

The paper is organized as follows. First, related work is presented in Section 2. The statistical framework of our novel grammar-based approach is described in Section 3. This framework derives two different models: the symbol likelihood that is described in Section 4, and the structural probability that is defined in Section 5. The parsing algorithm associated with this statistical framework is given in Section 6, and the experimentation carried out, along with analysis and discussion of the results, is reported in Section 7. Finally, conclusions and future work are presented in Section 8.

2. Related Work

The problem of automatic mathematical expression recognition has been studied for decades [11]. Many approaches have been proposed [12, 4, 13], but unfortunately most of them cannot be properly compared due to the lack of public datasets or standard metrics.

The Infty corpus [14] was released several years ago as a great resource of printed math expressions. More recently the MathBrush dataset [15] provided another important resource for handwritten math expression recognition. Over the last few years, the rapid growth of tactile devices and human-based interfaces has brought more attention to handwriting recognition solutions. With the recent editions of the CROHME competition [10], and the development of a set of performance evaluation metrics [16, 17], mathematical expression recognition has become a very active research field. In last three editions of the CROHME competition, systems were submitted from nine different countries.

Different approaches have been presented for math expression recognition. Zanibbi and Blostein [4] recognized an expression as a tree, and proposed a system based on a sequence of tree transformations. Eto and Suzuki [18] developed a model for printed math expression recognition that computed the minimum spanning tree of a network representation of the expression. Shi *et al.* [19, 20] presented a system where symbol segmentation and recognition were tackled simultaneously based on graphs. They then generated several symbol candidates for the best segmentation, and the recognized expression was computed in the final structural analysis [21].

Given the well-known structure of mathematical notation, many approaches are based on grammars because they constitute a natural way to model this problem. In fact, the first proposals on math expression recognition were grammar-based [11, 12]. Since then, different studies have been developed using different types of grammars. For instance, Chan and Yeung [22] used definite clause grammars, the Lavirotte and Pottier [23] model was based on graph grammars, Yamamoto *et al.* [24] presented a system using Probabilistic Context-Free Grammars (PCFG), and MacLean and Labahn [13] developed an approach using relational grammars and fuzzy sets. In this paper we will focus on models based on PCFG.

Proposals based on PCFG use grammars to model the structure of the expression, but the recognition systems are different. Garain and Chaudhuri [25] proposed a system that combines online and offline information in the structural analysis. First, they created online hypotheses based on determining baselines in the input expression, and then offline hypotheses using recursive horizontal and vertical splits. Finally they used a context-free grammar to guide the process of merging the hypotheses. Yamamoto *et al.* [24] presented a version of the CYK algorithm for parsing bidimensional PCFG (2D-PCFG) with the restriction that symbols and relations must follow the writing order. They defined probability functions based on a region representation called "hidden writing area". Průša and V. Hlaváč [26] described a system for offline recognition using 2D context-free grammars. Their proposal was penalty-based such that weights were associated with regions and syntactic rules. The model proposed by Awal *et al.* [5] considers several segmentation hypotheses based on spatial information, and the symbol classifier has a rejection class in order to avoid incorrect segmentations.

In this paper we present a formal model that is grounded in two studies. First, the system for math expression recognition based on parsing 2D-PCFG presented in [6]. That model tackled symbol segmentation by computing the connected components of the input strokes and merging them using productions of the grammar (e.g. an equals sign is a line below another line). However, this strategy required consideration of additional classes for symbol composition (e.g. an i without the dot or linked letters in functions) and finding proper spatial relations for their combination. Moreover, it could not account for touching symbols or symbols with segmentations that have not been taken into account with specific productions in the grammar (for instance, broken symbols like π in Fig. 5). Thus, segmentation was not a hidden variable but depended on design decisions of the grammar and symbol composition. Also, in [6] the estimation of the 2D-PCFG was not tackled.

Second, in order to overcome the problems of this segmentation methodology, we integrated a stroke-based approach similar to [19, 20] into the parsing process of 2D-PCFG. The solution presented in [19, 20] combined several stochastic sources of information on symbol segmentation, symbol recognition and symbol relationships in order to determine the best overall segmentation and recognition. However, it had the restriction that symbols must be written with consecutive strokes in time and structural analysis was performed as a decoupled step.

As a result, in this paper we develop a statistical framework for mathematical expression recognition which main contributions with regard to [6, 19, 20] are: i) a fully integrated approach based on 2D-PCFG using strokes as primitives with no time order assumptions. Our proposal integrates several stochastic information sources in order to globally determine the most likely mathematical expression. In this advanced framework, segmentation becomes a hidden variable; ii) we deal with the estimation of all the probabilistic sources of information and the reduction of the search space; iii) the framework is able to deal with all possible mathematical expressions provided that adequate models (mainly 2D-PCFG) are defined.

3. Statistical Framework

In online handwritten math expression recognition, the input is a sequence of strokes, and these strokes are in themselves a sequence points. Fig. 1 shows an example of the input for a mathematical expression. As you can see, the temporal sequence of strokes

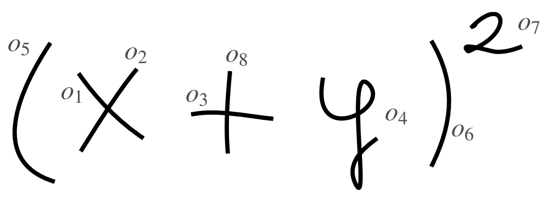


Figure 1: Example of input for an online handwritten math expression. The order of the input sequence of strokes is labeled ($\boldsymbol{o} = o_1 o_2 \ldots o_8$).

does not correspond necessarily to the sequence of symbols that it represents. For example, we can see that the user first wrote the subexpression $x - y$, then the user added the parentheses and its superscript $(x - y)^2$, finally converting the subtraction into an addition $(x + y)^2$. This example shows that some symbols might not be made up of consecutive strokes (e.g. the + symbol in Fig. 1). This means that the mathematical expression would not be correctly recognized if it was parsed monotonically with the input, i.e. processing the strokes in the order in which they were written. Meanwhile, the sequence of symbols that make up a subexpression does not have to respect the writing order (e.g. the parentheses and the subexpression they contain in Fig. 1).

Given a sequence of input strokes, the output of a mathematical expression recognizer is usually a sequence of symbols [19]. However, we consider that a significant element of the output is the structure that defines the relationship between the symbols making up the final mathematical expression. As mentioned above, we propose modeling the structural relationships of a mathematical expression using a statistical grammatical model. By doing so, we define the problem of mathematical expression recognition as obtaining the most likely parse tree given a sequence of strokes. Fig. 2 shows a possible parse tree for the expression given in Fig. 1, where we can observe that a (context-free) structural model would be appropriate due to, for instance, structural dependencies in bracketed expressions. The output parse tree represents the structure that relates all the symbols and subexpressions that make up the input expression. The parse tree derivation produces the sequence of pre-terminals that represent the recognized mathematical symbols. Furthermore, to generate this sequence of pre-terminals, we must take into account all stroke combinations in order to form the possible mathematical symbols.

Taking these considerations into account, two main problems have been observed. First, the segmentation and recognition of symbols is closely related to the alignment of mathematical symbols to strokes. Second, the structural analysis of a math expression addresses the problem of finding the parse tree that best accounts for the relationships between different mathematical symbols (pre-terminals). Obviously, these two problems are closely related. Symbol recognition is influenced by the structure of the

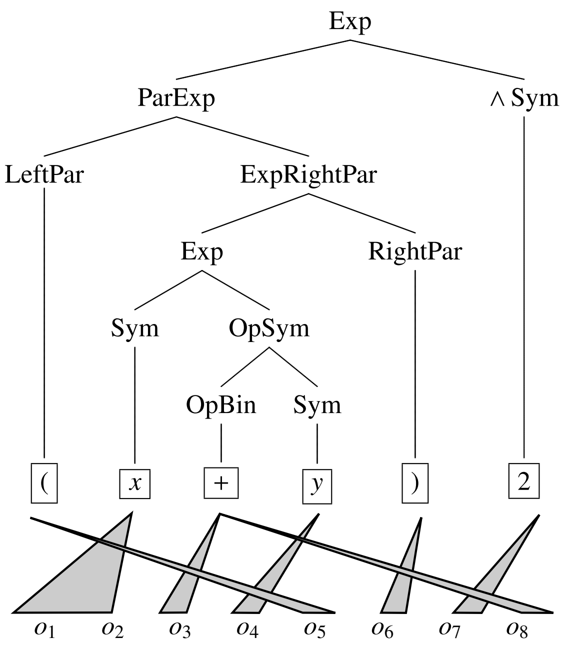


Figure 2: Parse tree of expression $(x + y)^2$ given the input sequence of strokes described in Fig. 1. The parse tree represents the structure of the math expression and it produces the 6 recognized symbols that account for the 8 input strokes.

mathematical expression, and detecting the structure of the math expression strongly depends on the segmentation and recognition of symbols. For these reasons we propose an integrated strategy that computes the most likely parse tree while simultaneously solving symbol segmentation, symbol recognition and the structural analysis of the input.

Formally, let a mathematical expression be a sequence of N strokes $\boldsymbol{o} = o_1 o_2 \dots o_N$. We pose the mathematical expression recognition as a structural parsing problem such that the goal is to obtain the most likely parse tree t that accounts for the input sequence of strokes $\boldsymbol{o}$:

$$\hat{t} = \arg\max_{t \in \mathcal{T}} \; p(t \mid \boldsymbol{o})$$

where $\mathcal{T}$ represents the set of all possible parse trees.

At this point, we consider the sequence of mathematical symbols $\boldsymbol{s} \in \mathcal{S}$ as a hidden variable, where $\mathcal{S}$ is the set of all possible sequences of symbols (pre-terminals) produced by the parse tree t: $\boldsymbol{s} = \text{yield}(t)$. This can be formalized as follows:

$$\hat{t} = \arg\max_{t \in \mathcal{T}} \sum_{\substack{\boldsymbol{s} \in \mathcal{S} \\ \boldsymbol{s} = \text{yield}(t)}} p(t, \boldsymbol{s} \mid \boldsymbol{o})$$

If we approximate the probability by the maximum probability parse tree, and assume that the structural part of the equation depends only on the sequence of pre-terminals

$\boldsymbol{s}$, the target expression becomes

$$\hat{t} \approx \arg\max_{t \in \mathcal{T}} \max_{\substack{\boldsymbol{s} \in \mathcal{S} \\ \boldsymbol{s} = \text{yield}(t)}} p(\boldsymbol{s} \mid \boldsymbol{o}) \cdot p(t \mid \boldsymbol{s}) \tag{1}$$

such that $p(\boldsymbol{s}|\boldsymbol{o})$ represents the observation (symbol) likelihood and $p(t|\boldsymbol{s})$ represents the structural probability.

This problem can be solved into two steps. First, by calculating the segmentation of the input into mathematical symbols and, second, by computing the structure that relates all recognized symbols [4].

However, in this study we propose a fully integrated strategy for computing Eq. (1) where symbol segmentation, symbol recognition and the structural analysis of the input expression are globally determined. This way, all the information is taken into account in order to obtain the most likely mathematical expression.

Next, in Section 4 we define the observation model that accounts for the probability of recognition and segmentation of symbols, $p(\boldsymbol{s}|\boldsymbol{o})$. The probability that accounts for the structure of the math expression $p(t|\boldsymbol{s})$ is described in Section 5.

4. Symbol Likelihood

As we have seen, in the recognition of online handwritten math expressions, the input is a sequence of strokes $\boldsymbol{o} = o_1 o_2 \ldots o_N$, which encodes a sequence of pre-terminals $\boldsymbol{s} = s_1 s_2 \ldots s_K$, $(1 \leq K \leq N)$ that represents the mathematical symbols. A symbol is made up of one or more strokes. Some approaches assumed that users always write a symbol with consecutive strokes [19, 24]. Although this assumption may be true in many cases, it constitutes a severe constraint that means that these models cannot account for symbols composed of non-consecutive written strokes. For example, the plus sign (+) in the expression in Fig. 1 is made up of strokes o_3 and o_8 and would not be recognized by a model that incorporated this assumption.

In this section we define a symbol likelihood model that is not based on time information but rather spatial information. This model is therefore able to recognize math symbols made up of non-consecutive strokes. Given a sequence of strokes, testing all possible segmentations could be unfeasible given the high number of possible combinations. However, it is clear that only strokes that are close together will form a mathematical symbol, which is why we tackle the problem using the spatial and geometric information available since, by doing so, we can effectively reduce the number of symbol segmentations considered. The application of this intuitive idea is detailed in Section 4.1.

Before defining the segmentation strategy adopted for modeling the symbol likelihood, we must introduce some preliminary formal definitions.

Definition 4.1. Given a sequence of N input strokes $\boldsymbol{o}$, and the set containing them $\text{set}(\boldsymbol{o}) = \{o_i \mid i : 1 \ldots N\}$, a *segmentation* of $\boldsymbol{o}$ into K segments is a partition of the set of input strokes

$$b(\boldsymbol{o}, K) = \{ b_i \mid i : 1 \ldots K \}$$

where each b_i is a set of (possibly non-consecutive) strokes representing a segmentation hypothesis for a given symbol.

Definition 4.2. We define $\mathcal{B}_K$ as the set of all possible segmentations of the input strokes $\boldsymbol{o}$ in K parts. Similarly, we define the set of all segmentations $\mathcal{B}$ as:

$$\mathcal{B} = \bigcup_{1 \le K \le N} \mathcal{B}_K$$

Once we have defined the set of all possible segmentations $\mathcal{B}$, we want to calculate the probability of the segmentation and recognition of symbols for a given sequence of strokes $\boldsymbol{o}$. In Eq. (1), we can define a generative model $p(\boldsymbol{s}, \boldsymbol{o})$, rather than $p(\boldsymbol{s}|\boldsymbol{o})$, because, given that the term $p(\boldsymbol{o})$ does not depend on the maximization variables $\boldsymbol{s}$ and t, we can drop it. The next step is to replace the sequence of N input strokes $\boldsymbol{o}$ by its previously defined set of segmentations, $b = b(\boldsymbol{o}, K) \in \mathcal{B}_K$ where $1 \le K \le N$. Finally, given K, we define a hidden variable that limits the number of strokes for each of the K pre-terminals (symbols) that make up the segmentation, $\boldsymbol{l} : l_1 \dots l_K$. Each l_i falls within the range $1 \le l_i \le \min(N, L_{\max})$, where $L_{\max}$ is a parameter that constrains the maximum number of strokes that a symbol can have.

$$p(\boldsymbol{s}, \boldsymbol{o}) = \sum_{1 \le K \le N} \sum_{b \in \mathcal{B}_K} \sum_{\boldsymbol{l}} p(\boldsymbol{s}, b, \boldsymbol{l})$$

In order to develop this expression, we factor it with respect to the number of pre-terminals (symbols) and assume the following constraints: 1) we approximate the summations by maximizations, 2) the probability of a possible segmentation depends only on the spatial constraints of the strokes it is made up of, 3) the probability of a symbol depends only on the set of strokes associated with it, and 4) the number of strokes for a pre-terminal depends only on the symbol it represents:

$$p(\boldsymbol{s}, \boldsymbol{o}) \approx \max_{K} \max_{b \in \mathcal{B}_K} \max_{\boldsymbol{l}} \prod_{i=1}^{K} p(b_i)\ p(s_i \mid b_i)\ p(l_i \mid s_i) \tag{2}$$

From Eq. (2) we can conclude that we need to define three models: a *symbol segmentation model*, $p(b_i)$, a *symbol classification model*, $p(s_i|b_i)$, and a *symbol duration model*, $p(l_i|s_i)$. Fig. 3 shows these models and how they are related. These three models are discussed in depth below.

4.1. Symbol Segmentation Model

Many symbols in mathematical expressions are made up of more than one stroke. For example, the symbols x and $+$ in Fig. 1 have two strokes, while symbols like π or $\neq$ usually require three strokes, etc. As we have already discussed, in this paper we are proposing a model where stroke segmentation is not based on temporal information, but rather on spatial and geometric information. We also defined $\mathcal{B}$ as the set of all possible segmentations. Given this definition of $\mathcal{B}$, it is easy to see that its size is exponential on the number of strokes N. In this section we first explain how to effectively reduce the number of segmentations considered, and then we describe the segmentation model used for computing the probability of a certain hypothesis $p(b_i)$.

Given a math expression represented by a sequence of strokes $\boldsymbol{o}$, the number of all possible segmentations $\mathcal{B}$ could be unfeasible. In order to reduce this set, we use two concepts based on geometric and spatial information: *visibility* and *closeness*. Let us first introduce some definitions.

Definition 4.3. The distance between two strokes o_i and o_j can be defined as the Euclidean distance between their closest points.

Definition 4.4. A stroke o_i is considered *visible* from o_j if the straight line between the closest points of both strokes does not cross any other stroke o_k.

If a stroke o_i is not *visible* from o_j we consider that their distance is infinite. For example, given the expression in Fig. 1, the strokes visible from o_4 are o_3, o_6 and o_8.

Furthermore, we know that a multi-stroke symbol is composed of strokes that are spatially close. For this reason, we only consider segmentation hypotheses b_i where strokes are close to each other.

Definition 4.5. A stroke o_i is considered *close* to another stroke o_j if their distance is shorter than a given threshold.

Using these definitions, we can characterize the set of possible segmentation hypotheses.

Definition 4.6. Let G be an undirected graph such that each stroke is a node and edges only connect strokes that are *visible* and *close*. Then, a segmentation hypothesis b_i is admissible if the strokes it contains form a connected subgraph in G.

Consequently, a segmentation $\boldsymbol{b}(\boldsymbol{o}, K) = b_1 b_2 \ldots b_K$ is admissible if each b_i is, in turn, admissible. These two geometric and spatial restrictions significantly reduce the number of possible symbol segmentations.

We need a segmentation model in order to calculate the probability that a given set of strokes (segmentation hypothesis, b_i) forms a mathematical symbol. Commonly, symbol segmentation models are defined using different features based on geometric information [27]. Also, the shape of the hypotheses has been used [28].

In this paper, we used a segmentation model very similar to the concept of *grouping likelihood* proposed in [19]. As in [19], we defined a set of geometric features associated with a segmentation hypothesis b_i. First, for each stroke o_j of b_i, we calculated the mean horizontal position, the mean vertical position and its size computed as the maximum value of horizontal and vertical size. Then, for each pair of strokes we calculated the difference between their horizontal positions, vertical positions and sizes. The average of these differences for each pair determined the features used for the segmentation model: *average horizontal distance* (d), *average vertical offset* (σ), and *average size difference* (δ). Additionally, we defined another feature: *average distance* (θ). This last feature is computed as the distance between the closest points of two strokes.

The authors in [19] used a scoring function such that these features were normalized using a fixed threshold value. However, this normalization depends on the resolution of the input. In order to overcome this restriction we normalized the features by the diagonal of the *normalized symbol size* (see Section 6.1), thereby ensuring that features are resolution-independent.

Finally, instead of the scoring function proposed in [19], we trained a Gaussian Mixture Model (GMM) using positive samples $c = 1$ (the strokes of b_i can form a math symbol) and a GMM using negative samples $c = 0$ (the strokes of b_i cannot form a math symbol) from the set of all admissible segmentations $\mathcal{B}$. A segmentation hypothesis b_i is represented by the 4-dimensional normalized feature vector $\mathrm{g}(b_i) = [d, \sigma, \delta, \theta]$, and the probability $p(b_i)$ that a hypothesis b_i forms a math symbol is obtained as

$$p(b_i) = p_{\mathrm{GMM}}(c = 1 \mid \mathrm{g}(b_i)) \tag{3}$$

4.2. Symbol Classification Model

Symbol classification is crucial in order to properly recognize mathematical notation. In this section we describe the model used for calculating the probability that a certain segmentation hypothesis b_i represents a math symbol s_i, i.e. the probability $p(s_i|b_i)$ required in Eq. (2).

Several approaches have been proposed in the literature to tackle this problem using different classifiers: Artificial Neural Networks [29], Support Vector Machines (SVM) [30], Gaussian Mixture Models (GMM) [19], elastic matching [31], Hidden Markov Models (HMMs) [32, 33] and Recurrent Neural Networks (RNN) [34]. Although not all of these approaches have been tested since some publications used private datasets, Bidirectional Long Short-Term Memory RNNs (BLSTM-RNN) are a state-of-the-art model that has outperformed previously reported results [34]. For this reason we used a BLSTM-RNN for mathematical symbol classification.

RNNs are a connectionist model containing a self-connected hidden layer. The recurrent connection provides information about previous inputs, meaning that the network can benefit from past contextual information [9]. Long Short-Term Memory (LSTM) is an advanced RNN architecture that allows cells to access context information over long periods of time. This is achieved by using a hidden layer made up of recurrently connected subnets called memory blocks [35].

Bidirectional RNNs [36] have two separate hidden layers that allow the network to access context information in both time directions: one hidden layer processes the input sequence forwards while another processes it backwards. The combination of bidirectional RNNs and the LSTM architecture results in BLSTM-RNNs, which have outperformed standard RNNs and HMMs in handwriting text recognition [35] and handwritten math symbol classification [34]. They are also faster than HMMs in terms of classification speed.

In order to train a BLSTM-RNN classifier, we computed several features from a segmentation hypothesis. Given a mathematical symbol represented as a sequence of points, for each point $p = (x, y)$ we extracted the following 7 online features:

- Normalized coordinates: (x, y) normalized values such that $y \in [0, 100]$ and the aspect-ratio of the sample is preserved.
- Normalized first derivatives: (x', y').
- Normalized second derivatives: (x'', y'').
- Curvature: k, the inverse of the radius of the curve at each point.

It should be noted that no resampling is required prior to the feature extraction process because first derivatives implicitly perform writing speed normalization [37].

Furthermore, the combination of online and offline information has been proven to improve recognition accuracy [32, 30, 38]. For this reason, we also rendered the image representing the symbol hypothesis b_i and extracted offline features to train another BLSTM-RNN classifier.

Following [38], for a segmentation hypothesis b_i, we generated the image representation as follows. We set the image height to H pixels and kept the aspect ratio (up to $5H$, in order to prevent creating too wide images). Then we rendered the image representation by using linear interpolation between each two consecutive points in a stroke. The final image was produced after smoothing it using a mean filter with a window sized 3×3 pixels, and binarizing for every pixel that is different from the background (white).

Given a binary image of height H and W columns, for each column we computed 9 offline features [7, 38]:

- Number of black pixels in the column.
- Center of gravity of the column.
- Second order moment of the column.
- Position of the upper contour of the column.
- Position of the lower contour of the column.
- Orientation of the upper contour of the column.
- Orientation of the lower contour of the column.
- Number of black-white transitions in the column.
- Number of black pixels between the upper and lower contours.

In order to classify a math symbol hypothesis, we trained two classifiers: a BLSTM-RNN with online feature vectors, and a BLSTM-RNN with offline feature vectors. The BLSTM-RNN was trained using a frame-based approach. Given a symbol hypothesis b_i of n frames, we computed a sequence of n feature vectors. Then, we obtained the posterior probability per symbol normalized as its average probability per frame:

$$p(s \mid b_i) = \frac{1}{n} \sum_{j=1}^{n} p(s \mid f_j) \tag{4}$$

Finally, given a segmentation hypothesis b_i and using Eq. (4), we obtained the posterior probability of a BLSTM-RNN with online features and the posterior probability of a BLSTM-RNN with offline features. We combined the probabilities of both classifiers using linear interpolation and a weight parameter (α). The final probability of the symbol classification model is calculated as

$$p(s_i \mid b_i) = \alpha \cdot p_{\text{on}}(s_i \mid b_i) + (1 - \alpha) \cdot p_{\text{off}}(s_i \mid b_i) \tag{5}$$

4.3. Symbol Duration Model

The symbol duration model accounts for the intuitive idea that a math symbol class is usually made up of a certain number of strokes. For example, the plus sign (+) is likely to be composed of two strokes, rather than one or more than two strokes.

As authors proposed in [19], a simple way to calculate the probability that certain symbol class s_i is made up of l_i strokes is

$$p(l_i \mid s_i) = \frac{c(s_i, l_i)}{c(s_i)} \tag{6}$$

where $c(s_i, l_i)$ is the number of times the symbol s_i was composed of l_i strokes and $c(s_i)$ is the total number of samples of class s_i in the set used for estimation. We smoothed these probabilities in order to account for unseen events.

5. Structural Probability

The statistical framework described in Section 3 defined the problem of recognizing a mathematical expression as finding the most likely parse tree t that accounts for the input strokes $\boldsymbol{o}$. Formally, the problem is stated in Eq. (1) such that two probabilities are required. In the previous section we presented the calculation of the *symbol likelihood* $p(\boldsymbol{s}|\boldsymbol{o})$. In this section we will define the *structural probability* $p(t|\boldsymbol{s})$ (see Fig. 3).

Although the most natural way to compute the most likely parse tree of an input sequence would be to define probabilistic *parsing models* $p(t|\boldsymbol{s})$, in the literature, this problem has usually been tackled using generative models $p(t, \boldsymbol{s})$ (*language models*) and, more precisely, *grammatical models* [39].

Next we define a generative model $p(t, \boldsymbol{s})$ based on a two-dimensional extension of the well-known context-free grammatical models.

5.1. 2D Probabilistic Context-Free Grammars

A context-free model is a powerful formalism able to represent the structure of natural languages. It is an appropriate model to account for math notation given the structural dependencies existing between the different elements in an expression (for instance, the parentheses in Fig. 1). We will use a two-dimensional extension of PCFG, a well-known formalism widely used for math expression recognition [11, 12, 24, 5, 6].

Definition 5.1. A *Context-Free Grammar* (CFG) G is a four-tuple $(\mathcal{N}, \Sigma, S, \mathcal{P})$, where $\mathcal{N}$ is a finite set of non-terminal symbols, Σ is a finite set of terminal symbols ($\mathcal{N} \cap \Sigma = \emptyset$), $S \in \mathcal{N}$ is the start symbol of the grammar, and $\mathcal{P}$ is a finite set of rules: $A \to \alpha$, $A \in N$, $\alpha \in (\mathcal{N} \cup \Sigma)^+$.

A CFG in Chomsky Normal Form (CNF) is a CFG in which the rules are of the form $A \to BC$ or $A \to a$ (where $A, B, C \in \mathcal{N}$ and $a \in \Sigma$).

Definition 5.2. A *Probabilistic CFG* (PCFG) $\mathcal{G}$ is defined as a pair (G, p), where G is a CFG and $p : P \to]0, 1]$ is a probability function of rule application such that $\forall A \in \mathcal{N} : \sum_{i=1}^{n_A} p(A \to \alpha_i) = 1$, where n_A is the number of rules associated with non-terminal symbol A.

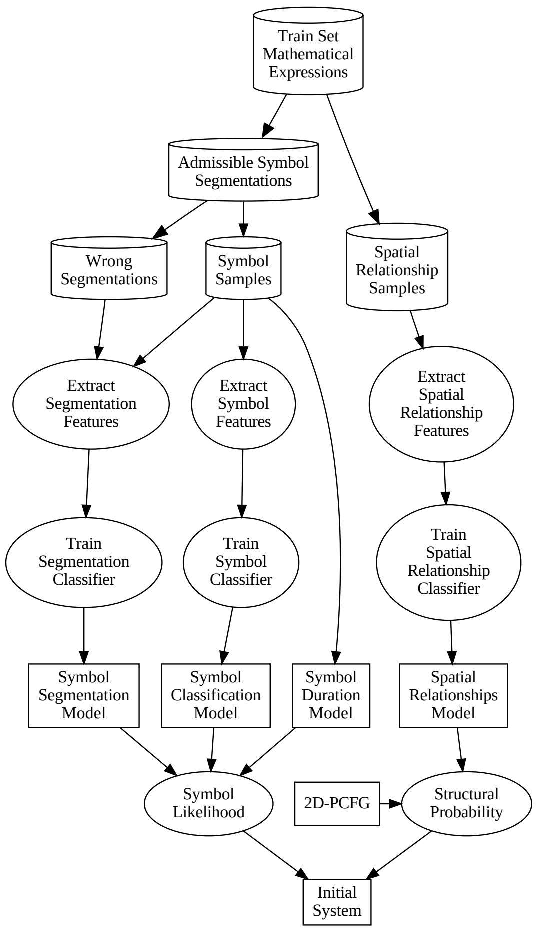

Train Set Mathematical Expressions
Admissible Symbol Segmentations
Wrong Segmentations
Symbol Samples
Spatial Relationship Samples
Extract Segmentation Features
Extract Symbol Features
Extract Spatial Relationship Features
Train Segmentation Classifier
Train Symbol Classifier
Train Spatial Relationship Classifier
Symbol Segmentation Model
Symbol Classification Model
Symbol Duration Model
Spatial Relationships Model
Symbol Likelihood
2D-PCFG
Structural Probability
Initial System

Figure 3: Diagram of the process for training the initial mathematical expression recognition system.

Definition 5.3. A *Two-Dimensional PCFG* (2D-PCFG) is a generalization of a PCFG, where terminal and non-terminal symbols describe two-dimensional regions. This grammar in CNF results in two types of rules: terminal rules and binary rules. First, the terminal rules $A \rightarrow a$ represent the mathematical symbols which are ultimately the terminal symbols of 2D-PCFG. Second, the binary rules $A \overset{r}{\rightarrow} BC$ have an additional parameter r that represents a given spatial relationship, and its interpretation is that regions B and C must be spatially arranged according to the spatial relationship r.

In Section 5.3 we will provide a full description of the spatial relationships considered here in order to address the recognition of mathematical expressions. The construction of the 2D-PCFG and the estimation of the probabilities are detailed in Section 7.3.5.

5.2. Parse Tree Probability

The 2D-PCFG model allows us to calculate the structural probability of a math expression in terms of the joint probability $p(t, \boldsymbol{s})$, such that in CNF is computed as:

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

where $p(\alpha|A)$ is the probability of the rule $A \to \alpha$ and represents the probability that α is derived from A. Moreover, $(A \to \alpha, t)$ denotes all rules $(A \to \alpha)$ contained in the parse tree t. In the defined 2D extension of PCFG, the composition of subproblems has an additional constraint according to a spatial relationship r. Let the spatial relationship r between two regions be a hidden variable. Then, the probability of a binary rule is written as:

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

When the inner probability in the previous addition is estimated from samples, the mode is the dominant term. Therefore, by approximating summations by maximizations, and assuming that the probability of a spatial relationship depends only on the subproblems B and C involved, the structural probability of a mathematical expression becomes:

$$p(t, \boldsymbol{s}) \approx \prod_{(A \to a, t)} p(a \mid A) \tag{7}$$

$$\prod_{(A \to BC, t)} \max_{r} \; p(BC \mid A) \; p(r \mid BC) \tag{8}$$

where $p(a|A)$ and $p(BC|A)$ are the probabilities of the rules of the grammar, and $p(r|BC)$ is the probability that regions encoded by non-terminals B and C are arranged according to spatial relationship r.

5.3. Spatial Relationships Model

The definition of Eq. (8) for computing the structural probability of a math expression requires a spatial relationship model. This model provides the probability $p(r|BC)$ that two subproblems B and C are arranged according to spatial relationship r.

A common approach for obtaining a spatial relationship model is to define a set of geometric features to train a statistical classifier. Most proposals in the literature define geometric features based on the bounding boxes of the regions [4, 6, 5], although a proposal based on shape descriptors has also been studied [40]. The geometric features are usually modeled using Gaussian models [5], SVM [6] or fuzzy functions [41], though some authors manually define specific functions [4, 24, 13].

In this work, we deal with the recognition of math expressions using six spatial relationships: *right* (BC), *below* ($\frac{B}{C}$), *subscript* (B_C), *superscript* (B^C), *inside* ($\sqrt{C}$) and *mroot* ($\sqrt[C]{\;}$).

In order to train a statistical classifier, given two regions B and C we define nine geometric features based on their bounding boxes [40] (see Fig. 4). This way, we compute the feature vector $\mathrm{h}(B, C)$ that represents their relationship and can be used for classification. The features are defined in Fig. 4, where H is the height of region C, feature D is the difference between the vertical centroids, and dhc is the difference between the horizontal centers. The features are normalized by the combined height of regions B and C.

The most challenging classification is between classes *right*, *subscript* and *superscript* [6, 40]. An important feature for distinguishing between these three relationships is the difference between vertical centroids (D). Some symbols have ascenders, descenders or certain shapes such that the vertical centroid is not the best placement for the symbol center.

With a view to improving the placement of vertical centroids, we divided symbols into four typographic categories: ascendant (e.g. d or λ), descendant (p, μ), normal ($x, +$) and middle ($7, \Pi$). For *normal* symbols the centroid is set to the vertical centroid. For *ascendant* symbols the centroid is shifted downward to $(centroid + bottom)/2$. Likewise, for *descendant* symbols the centroid is shifted upward to $(centroid + top)/2$. Finally, for *middle* symbols the vertical centroid is defined as $(top + bottom)/2$.

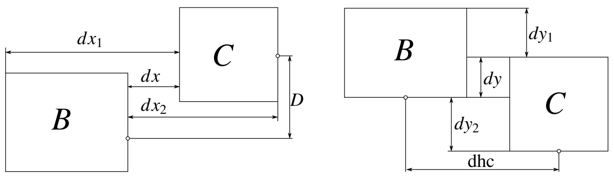
Once we defined the feature vector representing a spatial relationship, we can train a GMM using labeled samples such that the probability of the spatial relationship model can be computed as the posterior probability provided by the GMM for class r

$$p(r \mid BC) = p_{\text{GMM}}(r \mid \text{h}(B, C))$$

This model is able to provide a probability for every spatial relationship r between any two given regions. However, there are several situations where we would not want the statistical model to assign the same probability as in other cases. Considering the expression in Fig. 5, the GMMs might yield a high probability for *superscript* relationship '3^x', for the *below* relationship '$\frac{\pi}{2}$', and for the *right* relationship '2 3'; though we might expect a lower probability, since they are not the true relationships in the correct math expression.

Intuitively, those symbols or subexpressions that are closer together should be combined first. Furthermore, two symbols or subexpressions that are not *visible* from each other should not be combined. These ideas are introduced into the spatial relationship model as a penalty based on the distance between strokes.

Specifically, given the combination of two hypotheses B and C, we computed a



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

Figure 4: Geometric features for classifying the spatial relationship between regions B and C.

penalty function based on the minimum distance between the strokes of B and C

$$\mathrm{penalty}(B, C) = 1/(\ 1 + \min_{o_i \in B,\, o_j \in C} \mathrm{d}(o_i, o_j)\ )$$

such that it is in the range $[0, 1]$. It should be noted that, although it is a penalty function, since it multiplies the probability of a hypothesis, the lower is the penalty value the more the probability is penalized.

This function is based on the single-linkage hierarchical clustering algorithm [42] where, at each step, the two clusters separated by the shortest distance are combined. We defined a penalty function in order to avoid making hard decisions, because it is not always the case that the two closest strokes must be combined first.

The final statistical spatial relationship probability is computed as the product of the probability provided by the GMM and the penalty function based on hierarchical clustering

$$p(r \mid BC) = p_{\mathrm{GMM}}(r \mid \mathrm{h}(B, C)) \cdot \mathrm{penalty}(B, C) \tag{9}$$

An interesting property of the application of the penalty function is that, given that the distance between non-visible strokes is considered infinite, this function prunes many hypotheses. Furthermore, it favors the combination of closer strokes over strokes that are further apart. For example, the superscript relationship between symbols 3 and x in Fig. 5, although it could be likely, the penalty will favor that the 3 is first combined with the fraction bar, and later the fraction bar (and the entire fraction) with the x.

6. Parsing Algorithm

In this section we present the parsing algorithm for mathematical expression recognition that maximizes Eq. (1). We define a CYK-based algorithm for 2D-PCFGs in the statistical framework described in this paper. Using this algorithm, we compute the most likely parse tree according to the proposed model.

The parsing algorithm is essentially a dynamic programming method. First, the initialization step computes the probability of several mathematical symbols for each possible segmentation hypothesis. Second, the general case computes the probability of combining different hypotheses such that it builds the structure of the mathematical expression.

The dynamic programming algorithm computes a probabilistic parse table γ. Following a notation similar to [43], each element of γ is a probabilistic non-terminal

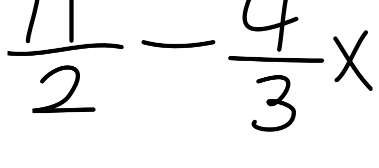


Figure 5: Example for hierarchical clustering penalty.

vector, where their components are defined as:

$$\gamma(A, b, l) \; = \; \hat{p}(A \overset{*}{\Rightarrow} b); \qquad l \; = | \, b \, |$$

such that $\gamma(A, b, l)$ denotes the probability of the best derivation that the non-terminal A generates a set of strokes b of size l.

Initialization: In this step the parsing algorithm computes the probability of every admissible segmentation $b \in \mathcal{B}$ as described in Section 4.1. The probability of each segmentation hypothesis is computed according to Eqs. (1) and (2) as

$$\gamma(A, b_i, l) = \max_{s} \{ \, p(s \mid A) \; p(b_i) \; p(s \mid b_i) \; p(l \mid s) \, \} \tag{10}$$

$$\forall A, \forall K, \; \forall b \in \mathcal{B}_K, \; 1 \leq i \leq |b|, \; 1 \leq l \leq \min(N, L_{\max})$$

where $L_{\max}$ is a parameter that constrains the maximum number of strokes that a symbol can have.

This probability is the product of a range of factors such that it is maximized for every mathematical symbol class s: probability of terminal rule, $p(s|A)$ (Eq. (7)), probability of segmentation model, $p(b)$ (Eq. (3)), probability of mathematical symbol classifier, $p(s|b)$ (Eq. (5)), and probability of duration model probability, $p(l|s)$ (Eq. (6)).

General case: In this step the parsing algorithm computes a new hypothesis $\gamma(A, b, l)$ by merging previously computed hypotheses from the parsing table until all N strokes are parsed. The probability of each new hypothesis is calculated according to Eqs. (1) and (8) as:

$$\gamma(A, b, l) = \max\{ \; \gamma(A, b, l), \; \max_{B,C} \; \max_{r} \; \max_{b_B, b_C} \{$$
$$p(BC \mid A) \gamma(B, b_B, l_B) \, \gamma(C, b_C, l_C) \, p(r \mid BC) \; \}\} \tag{11}$$
$$\forall A, \; 2 \leq l \leq N$$

such that $b = b_B \cup b_C$; $b_B \cap b_C = \emptyset$ and $l = l_B + l_C$.

This expression shows how a new hypothesis $\gamma(A, b, l)$ is built by combining two subproblems $\gamma(B, b_B, l_B)$ and $\gamma(C, b_C, l_C)$, considering both syntactic and spatial information: probability of binary grammar rule $p(BC|A)$ (Eq. (8)) and probability of spatial relationship classifier $p(r|BC)$ (Eq. (9)). It should be noted that both distributions significantly reduce the number of hypotheses that are merged. Also, the probability is maximized taking into account that a probability might already have been set by the Eq. (10) during the initialization step.

Finally, the most likely hypothesis and its associated derivation tree $\hat{t}$ that accounts for the input expression can be retrieved in $\gamma(S, \boldsymbol{o}, N)$ (where S is the start symbol of the grammar).

6.1. Complexity and Search Space

We have defined an integrated approach for math expression recognition based on parsing 2D-PCFG. The dynamic programming algorithm is defined by the corresponding recursive equations. The initialization step is performed by Eq. (10), while the general case is computed according to Eq. (11). In addition to the formal definition, there are some details of the parsing algorithm regarding the search space that need further explanation.

Once several symbol hypotheses have been created during the initialization step, the general case is the core of the algorithm where hypotheses of increasing size $2 \leq l \leq N$ are generated with Eq. (11). For a given size l, we have to test all the sizes in order to split l into hypotheses b_B and b_C such that $l = l_B + l_C$. Once the sizes are set, for every set of strokes b_B we have to test every possible combination with another set b_C using the binary rules of the grammar $A \xrightarrow{r} BC$. According to this, we can see that the time complexity for parsing an input expression of N strokes is $O(N^4|P|)$ where $|P|$ is the number of productions of the grammar. However, this complexity can be reduced by constraining the search space.

The intuitive idea is that, given a set of strokes b_B, we do not need to try to combine it with every other set b_C. A set of strokes b_B defines a region in space, allowing us to limit the set of hypothesis b_C to those that fall within a region of interest. For example, given symbol 4 in Fig. 5, we only have to check for combinations with the fraction bar and symbol 3 (below relationship) and the symbol x (right or sub/superscript relationships).

We applied this idea as follows. Given a stroke o_i we define its associated region $r(o_i) = (x, y, s, t)$ in the 2D space as the minimum bounding box that contains that stroke, where (x, y) is the top-left coordinate and (s, t) the bottom-right coordinate of the region. Likewise, given a set of strokes $b = \{o_j \mid 1 \leq j \leq |b|\}$ we define $r(b) = (x_b, y_b, s_b, t_b)$ as the minimum rectangle that contains all the strokes $o_j \in b$. Therefore, given a spatial region $r(b_B)$ we retrieve only the hypotheses b_C whose region $r(b_C)$ falls in a given area $\mathcal{R}$ relative to $r(b_B)$. Fig. 6 shows the definition of the regions in the space in order to retrieve relevant hypotheses to combine with b_B depending on the spatial relation. The dimensions of the normalized symbol size (R_w, R_h) are computed as: R_w, the maximum between the average and median width of the input strokes; and R_h, the maximum between the average and median height of the input strokes. These calculations are independent of the input resolution. The normalized symbol size is also used to normalize other distance-related metrics in the model, like determining what strokes are close together in the multi-stroke symbol recognition or the normalization factor of features in the segmentation model.

In order to efficiently retrieve the hypotheses falling in a given region $\mathcal{R}$, every time a set of hypotheses of size l_A is computed, we sort this set according to the x coordinate of every region $r(b_A)$ associated with $\gamma(A, b_A, l_A)$. This sorting operation has cost $O(N \log N)$. Afterwards, given a rectangle $r(b_B)$ in the search space and a size l_C, we can retrieve the hypotheses $\gamma(C, b_C, l_C)$ falling within that area by performing a binary search over that set in $O(\log N)$. Although the regions are arranged in two-dimensions and they are sorted only in one dimension, this approach is reasonable since math expressions grow mainly from left to right.

Assuming that this binary search will retrieve a small constant number of hypothesis, the final complexity achieved is $O(N^3 \log N|P|)$. Furthermore, many unlikely hypotheses are pruned during the parsing process.

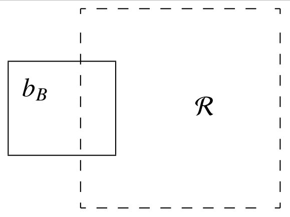
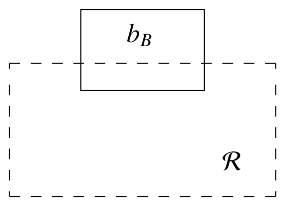
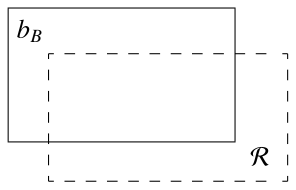
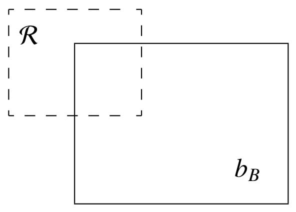
Right, Sub/Superscript $x = \max(r(b_B).x + 1,$ $r(b_B).s - R_w)$ $y = r(b_B).y - R_h$ $s = r(b_B).s + 8R_w$ $t = r(b_B).t + R_h$	
Below $x = r(b_B).x - 2R_w$ $y = \max(r(b_B).y + 1,$ $r(b_B).t - R_h)$ $s = r(b_B).s + 2R_w$ $t = r(b_B).t + 3R_h$	
Inside $x = r(b_B).x + 1$ $y = r(b_B).y + 1$ $s = r(b_B).s + R_w$ $t = r(b_B).t + R_h$	
Mroot $x = r(b_B).x - 2R_w$ $y = r(b_B).y - R_h$ $s = \min(r(b_B).s,$ $r(b_B).x + 2R_w)$ $t = \min(r(b_B).t,$ $r(b_B).y + 2R_h)$	

Figure 6: Spatial regions defined to retrieve hypotheses relative to hypothesis b_B according to different relations.

7. Experimentation

We have developed `seshat`[1], an open-source system that implements the parsing algorithm of the approach proposed in this study[2]. In order to evaluate the performance of this proposal, we carried out several experiments using a large public dataset from a recent international competition [10].

7.1. Dataset

The CROHME 2013 competition [10] released a large resource for math expression recognition as a result of combining and normalizing several datasets. Currently, it represents the largest public dataset of online handwritten math expressions. This dataset has 8 836 math expressions for training and a test set of 671 math expressions. The number of math symbol classes is 101.

We report results on the competition test set in order to provide comparable results. We needed a validation set for the estimation of the models used in the parsing algorithm. For this reason, we extracted 500 math expressions from the training set, such that both sets had the same distribution of symbols. Therefore, the sets used in the experimentation described in this paper were a training set made up of $8,336$ expressions (81K symbols), a validation set with 500 expressions (5K symbols) and the CROHME 2013 test set containing 671 math expressions (6K symbols).

7.2. Evaluation

Automatic performance evaluation of math expression recognition is not straightforward [44, 45]. Problems arise resulting from ambiguity in the ground-truth representation in that several encodings can account for the same expression [46]. Also, there are different sources of error: segmentation, symbol recognition and structural errors; and a global value is usually desired to measure a recognition result.

The set of metrics based on label graphs [16, 17] is intended to overcome the ambiguities in the ground-truth representation. The label graph representation uses strokes as primitives to encode a math expression. A label graph is essentially a directed graph over primitives that can be represented using adjacency matrices. In a label graph, nodes represent strokes, while edges define segmentation and spatial relationships. Computing Hamming distances over label graphs allows classification, segmentation and structural errors to be characterized separately, or using a single measure.

The last CROHME competitions and publications in the field report this set of metrics [10]. Consequently, we report results using them. The set of metrics includes precision and recall at symbol level, and Hamming distances at stroke level. Furthermore, two global error metrics are provided: ΔB_n which is the Hamming distance between the adjacency matrices normalized by the label graph length; and ΔE is the average per-stroke classification, segmentation and layout errors [16, 17].

[1]`https://github.com/falvaro/seshat`

[2]Demo available at `http://cat.prhlt.upv.es/mer`

7.3. Estimation of Models

The proposal presented in this paper has several probabilistic models that are used as a source of information during the recognition process of a math expression. Below we outline how each model is estimated. Fig. 3 shows the probabilistic models and processes used for learning them.

7.3.1. Symbol Classification Model

We trained two BLSTM-RNN using the RNNLIB library[3], one for online classification and the other for offline classification. We used the same configuration for both BLSTM-RNN. The only difference was the size of the input layer, which was determined by the feature set: 7 inputs for online data and 9 inputs for offline data. The output layer size was 101, i.e. the number of symbol classes, and the forward and backward hidden layers each contained 100 LSTM memory blocks.

The network weights were initialized with a Gaussian distribution of mean 0 and standard deviation 0.1. The network was trained using online gradient descent with a learning rate of 0.0001 and a momentum of 0.9. This configuration obtained good results in both handwritten text recognition [35] and handwritten math symbol classification [34]. We trained each network until the error ceased to improve on the validation set for 50 epochs.

7.3.2. Symbol Segmentation Model

Given the expressions of the training set, we extracted all admissible segmentation hypotheses $b_i \in \mathcal{B}$(see Fig. 3). Thus, we obtained 360K 4-dimensional samples of symbol segmentations, where 7.5% samples corresponded to proper symbol segmentations and 92.5% samples were incorrect segmentations. From the validation set we extracted 35K samples: 4.4% correct and 95.6% incorrect segmentations.

We trained the GMMs using the training samples and the Expectation-Maximization algorithm. The parameters of the model were chosen by minimizing the error when classifying the segmentation hypotheses of the validation set. The number of mixtures in the final GMMs was 5.

7.3.3. Symbol Duration Model

To estimate the duration model we used a simple ratio between the number of times a given symbol s_i was composed of l_i strokes and the total number of samples of that symbol found in the training set. The values for a math symbol class were smoothed using add-one smoothing [47] in the range $[1, L_{\max}]$.

7.3.4. Spatial Relationships Model

We extracted the spatial relationships between symbols and subexpressions for all the math expressions of the training set (see Fig. 3). This extraction had to be carried out taking into account the centroid computation for each type of symbol and the combination of regions. By doing so, we obtained 68K 9-dimensional samples of spatial

[3]`http://sourceforge.net/projects/rnnl/`

relationships for training and 4K samples for validation. The distribution of the classes was quite unbalanced. In the training set the *right* relationships were very common (71.84%), while *inside* and *mroot* relationships were infrequent (2.51% and 0.19%). The percentage of samples of *below*, *subscript* and *superscript* were about 12.08%, 5.56% and 7.82%, respectively.

Once we had the set of training samples, we trained a GMM per class using the Expectation-Maximization algorithm. The number of mixtures of the final GMMs was 5, since it represented the best trade-off between the complexity of the model and the classification error in the validation set.

7.3.5. 2D-PCFG Estimation

The structure of the mathematical expressions is described by a 2D-PCFG. Since the rules of math notation are well-known, starting from the CFG provided by the organizers of the CROHME competition, we manually modified it to improve the modeling of some structures. We also added productions that increase ambiguity in order to model certain structures like the relations between categories of symbols (uppercase/lowercase letters, numbers, etc.). However, the probabilities of the productions of the 2D-PCFG have to be estimated.

An usual approach to estimating probabilistic grammars is the Viterbi score [48]. We recognized the expressions of the training set in order to obtain the most likely derivation trees according to the grammar. As recognizing the training set will introduce errors into the computed trees, we used constrained parsing [46] to obtain the parse tree that best represents each training sample. Fig. 7 shows an scheme of the this training process. Then, the probability of a production $A \to \alpha$ was calculated as

$$p(A \to \alpha) = \frac{c(A \to \alpha)}{c(A)}$$

such that $c(A \to \alpha)$ is the number of times that the rule $A \to \alpha$ was used when recognizing the training set, and $c(A)$ is the total number of productions used that have A as left-hand operator. In order to account for unseen events, we smoothed the probabilities using add-one smoothing [47].

7.4. Parameter Setting

The parsing algorithm defined in Section 6 has two steps. First, when the parsing table is initialized, multi-stroke symbol hypotheses are introduced as subproblems of $1 \le n \le L_{\max}$ strokes following Eq. (10). Then, in the general case, the parsing table is filled with hypotheses of $n \ge 2$ strokes by combining smaller subproblems using Eq. (11). This produces a scaling problem of the probabilities.

The probability of a hypothesis of size $n \ge 2$ created in the initialization step is the product of four probabilities, while the hypothesis resulting from the combination of subproblems will involve $6n - 2$ terms in the calculation. Moreover, the different probabilistic distributions have been estimated separately, which leads to values in different scales.

For these reasons, following [20], we assigned different exponential weights to each model probability, and we also added an insertion penalty in the initialization

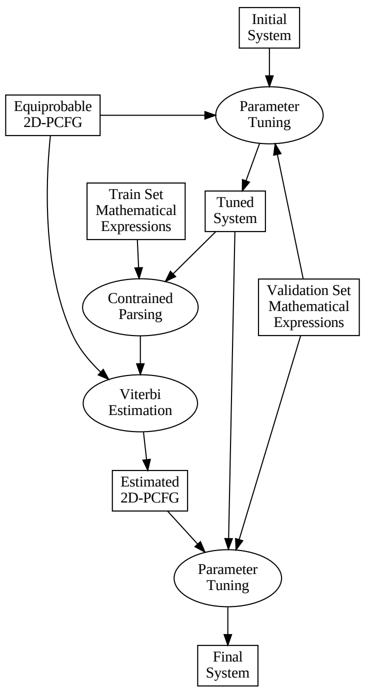


Figure 7: Diagram of the process for training the final mathematical expression recognition system.

step (Eq. (10)). These parameters alleviate the scaling differences of the probabilities. Furthermore, the weights help to adjust the contribution of each model to the final probability, since some sources of information are more relevant than the others.

The parameters of the parsing algorithm are: insertion penalty, exponential weights, segmentation distance threshold and maximum number of strokes per symbol ($L_{\max}$). We set $L_{\max} = 4$ because it accounts for 99.81% of the symbols in the dataset. The remaining parameters were set initially to 1.0 and we tuned them using the downhill simplex algorithm [49] minimizing the ΔE metric [16] when recognizing the validation set. Fig. 7 shows this process as the final process.

7.5. Experiments and Results

After training all the models and parameters using the training set, we used the system that obtained the best results on the validation set to classify the test set of the CROHME 2013 competition. Eight systems participated, including a preliminary version of this model (system IV). All but two of the systems used only the competition training set ($8,836$ math expressions). System III also used 468 additional math expressions, and System VII was trained using roughly $30,000$ math expressions from a private corpus. The description of each system can be found in [10].

Table 1 shows the performance metrics at symbol level, and Table 2 shows results at stroke level. Results show that system VII performed the best, obtaining very good

Table 1: Object-level evaluation for the CROHME'13 dataset. Systems sorted by decreasing recall for correct symbol segmentation and classification (Seg+Class).

	Segments		Seg+Class		DAG Relations	
	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.
VII	97.9	98.1	93.0	93.3	95.2	95.5
seshat	**92.0**	**90.7**	**82.2**	**81.0**	**88.0**	**82.0**
IV*	85.0	87.1	73.9	75.8	76.3	79.9
VIII	90.3	86.9	73.8	71.0	73.0	77.7
V	84.5	86.5	66.7	68.3	72.6	74.3
II	80.7	86.4	66.4	71.1	45.8	63.0
III	85.2	77.9	62.6	57.3	88.5	78.3
VI	57.9	47.3	47.7	39.0	31.8	70.0
I	46.9	38.4	25.2	20.6	33.7	71.6

results. It was awarded the strongest system in the competition. However, as they used a large private dataset we were not able to fairly compare its performance to that of the other systems. System IV was a preliminary version of `seshat` and was awarded the best system trained using only the CROHME training dataset. The main differences between System IV and `seshat` are as follows: `seshat` includes offline information in the symbol classifier; symbol segmentation and spatial relationships classification are carried out by GMMs in `seshat` and by SVMs in system IV; `seshat` uses *visibility* between strokes and a clustering-based penalty; and the probabilities of the grammar were not estimated in System IV.

Table 2: Stroke-level evaluation for the CROHME'13 dataset. Systems sorted by increasing ΔE. ΔB_n and ΔE are measured on directed labels graphs (8548 strokes; 81007 (undirected) stroke pairs).

	Label Hamming Distances				μ error (%)	
	Strokes	Pairs	Seg	Rel	ΔB_n	ΔE
VII	537	1 777	170	1 607	2.4	4.3
seshat	**1 583**	**7 829**	**700**	**7 129**	**6.9**	**13.2**
IV*	2 187	9 493	1 201	8 292	10.1	18.3
VIII	2 302	15 644	4 945	10 699	12.1	19.3
II	2 748	19 768	1 527	18 241	13.9	22.0
V	2 898	10 803	1 228	9 575	12.7	22.8
III	3 415	15 135	1 262	13 873	15.0	26.2
VI	4 768	43 893	5 094	38 799	27.6	36.7
I	6 543	41 295	5 849	35 446	26.8	41.6

The system presented in this paper significantly outperformed at all levels the other systems that were trained using the CROHME dataset. At symbol level, symbol classification of correctly segmented symbols of seshat obtained recall and precision of about 82.2% and 81.0%, while the next best system (system VIII) obtained 73.8 and 71.0%. The absolute difference was more than 8% in recall and 10% in precision. Regarding spatial relationships, recall and precision for seshat stood at 88% and 82%, while System VIII gave 73% and 77.7%. This translates into an absolute difference of about 14% in recall and 4.3% in precision. Results at stroke-level were also better than those of the systems trained only using the CROHME dataset. The systems were ranked according to the global error metric ΔE, where seshat had 13.2%, some 6.1% less than next best system (19.3). Table 3 shows that the confidence interval [50] was 13.2% ± 0.9 at 95% of confidence. We were not able to obtain confidence interval for the other systems because the final system outputs were not freely available.

In addition to the experimentation comparing our proposed model to other systems, it is interesting to see how each of the stochastic sources contribute to overall system performance. For this reason we also carried out an experiment to observe this behavior. Some models are mandatory for recognizing math expressions: the symbol classifier, the spatial relationships classifier and the grammar. We performed an experiment using only these models (base system), then adding the remaining models one by one. Also, the grammar initially had equiprobable productions and then we compared the performance when the probabilities of the rules were estimated.

Table 3 shows the changes on system performance when each source of information is added. Global error metrics ΔB_n and ΔE consistently decreased with each added feature. Confidence intervals computed for ΔE showed that the improvements were significant from the first row to the last row. Symbol segmentation and symbol recognition also improved with each addition. It is interesting that, when no segmentation model was used, symbol segmentation still gave good results. This was the case because the parameters of the integrated approach converged to high values of the insertion penalty and low values of the segmentation distance threshold. In this way, the parameters of the system itself could alleviate the lack of a segmentation model. In any case, when the segmentation model was included, system performance improved significantly. Furthermore, we would like to remark that, when the relations penalty was included in the model, the number of hypotheses explored was reduced by 56.7%.

Table 3: Contribution to overall system performance of the different sources of information used. The models and features listed in each row are cumulative, such that the system shown in the last row includes all information sources. Confidence intervals [50] were computed for ΔE.

	Segments		Seg+Class		Relations		Label Hamming Distances				μ error (%)	
	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Strokes	Pairs	Seg	Rel	ΔB_n	ΔE
Base system	87.6	83.6	78.3	74.7	87.0	75.0	1 987	11 999	1 172	10 827	9.1	16.5 ± 1.0
+ duration mod.	88.0	84.2	78.7	75.3	86.2	75.5	1 928	12 055	1 138	10 917	9.0	15.8 ± 1.1
+ segment. mod.	90.9	90.4	81.3	80.8	81.5	73.7	1 634	10 272	834	9 438	7.6	13.7 ± 1.0
+ rel. penalty	91.8	91.3	82.0	81.5	81.4	74.0	1 556	10 397	724	9 673	7.5	13.2 ± 0.9
+ gram. estim.	92.0	90.7	82.2	81.0	88.0	82.0	1 583	7 829	700	7 129	6.9	13.2 ± 0.9

The structural analysis is harder to evaluate. Prior to estimating the grammar probabilities, the results at object level seem to worsen when the segmentation model was included, although at stroke-level the errors in spatial relationships decreased from about $11,000$ to $9,500$ stroke pairs. Because of the inheritance of spatial relations in label graphs [16] some types of structural errors can produce more stroke-pair errors than others [17]. Specifically, when the segmentation model was used, the segmentation distance threshold was approximately twice as high as the value of the threshold in the base system. This had two effects. First, that the system was able to account for more symbol segmentations, as shown by the corresponding metrics. Second, that a bad decision in symbol segmentation can lead to worse structural errors. Nevertheless, the estimation of the probabilities of the grammar led to great improvements in the detection of the structure of the expression with barely any changes in symbol recognition performance.

8. Conclusions

In this paper we have presented a statistical model for online handwritten math expression recognition. We defined the statistical framework of a system based on parsing 2D-PCFG that integrates several sources of stochastic information. This constitutes a fully integrated approach that simultaneously determines the best symbol segmentation, symbol recognition and the most likely structure of the recognized mathematical expression. The search space is restricted with spatial and geometric information for making the search feasible.

We developed the software that implements the proposed system and released it as open-source. Furthermore, we report results on a large public dataset used in an international competition. A detailed discussion of system performance and the contribution of each part of the model is also provided. A preliminary version of this model was awarded as best system using only the training dataset of the CROHME competition [10]. Finally, the model presented in this study significantly outperforms other approaches at all levels.

The estimation of the 2D-PCFG had an important effect on system's performance. Since the estimation of a PCFG requires many data, we need to obtain more resources for this purpose.

One of the limitations of the system is that for giving account of unseen mathematical expression, the 2D-PCFG should be adequately smoothed or an error-correcting technique should be used. Also, each source of information could be improved with other features and classifiers. Moreover, this model deals with online handwritten math expression recognition, and is based on strokes as primitives. However, the proposed model is directly applicable to offline expressions, considering connected components as primitives. Future work will be focused on applying this model to offline mathematical expression recognition, both printed and handwritten.

Acknowledgments

This work was partially supported by the Spanish MINECO under the STraDA research project (TIN2012-37475-C02-01) and the FPU grant (AP2009-4363).

References

[1] K. Chan, D. Yeung, Mathematical expression recognition: a survey, International Journal on Document Analysis and Recognition 3 (2000) 3–15.

[2] R. Zanibbi, D. Blostein, Recognition and retrieval of mathematical expressions, International Journal on Document Analysis and Recognition 15 (4) (2012) 331–357.

[3] R. Plamondon, S. Srihari, Online and off-line handwriting recognition: a comprehensive survey, Pattern Analysis and Machine Intelligence, IEEE Trans. on 22 (1) (2000) 63–84.

[4] R. Zanibbi, D. Blostein, J. Cordy, Recognizing mathematical expressions using tree transformation, IEEE Trans. on Pattern Analysis and Machine Intelligence 24 (11) (2002) 1–13.

[5] A.-M. Awal, H. Mouchère, C. Viard-Gaudin, A global learning approach for an online handwritten mathematical expression recognition system, Pattern Recognition Letters 35 (0) (2014) 68 – 77.

[6] F. Álvaro, J.-A. Sánchez, J.-M. Benedí, Recognition of on-line handwritten mathematical expressions using 2d stochastic context-free grammars and hidden markov models, Pattern Recognition Letters 35 (0) (2014) 58 – 67.

[7] U.-V. Marti, H. Bunke, Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system, International Journal of Pattern Recognition and Artificial Intelligence 15 (01) (2001) 65–90.

[8] A. H. Toselli, et al., Integrated Handwriting Recognition and Interpretation using Finite-State Models, Int. Journal on Pattern Recognition and Artificial Intelligence 18 (4) (2004) 519–539.

[9] B. Pearlmutter, Learning state space trajectories in recurrent neural networks, in: International Joint Conference on Neural Networks, Vol. 2, 1989, pp. 365–372. doi:10.1109/IJCNN.1989.118724.

[10] H. Mouchère, C. Viard-Gaudin, R. Zanibbi, U. Garain, D. H. Kim, ICDAR 2013 CROHME: Third International Competition on Recognition of Online Handwritten Mathematical Expressions, in: International Conference on Document Analysis and Recognition, 2013, pp. 1428–1432.

[11] R. H. Anderson, Syntax-directed recognition of hand-printed two-dimensional mathematics, in: ACM Symposium on Interactive Systems for Experimental Applied Mathematics, 1967, pp. 436–459.

[12] P. A. Chou, Recognition of equations using a two-dimensional stochastic context-free grammar, in: Visual Communications and Image Processing IV, Vol. 1199, 1989, pp. 852–863.

[13] S. MacLean, G. Labahn, A new approach for recognizing handwritten mathematics using relational grammars and fuzzy sets, International Journal on Document Analysis and Recognition 16 (2) (2013) 139–163.

[14] M. Suzuki, S. Uchida, A. Nomura, A ground-truthed mathematical character and symbol image database, in: International Conference on Document Analysis and Recognition, 2005, pp. 675–679.

[15] S. MacLean, G. Labahn, E. Lank, M. Marzouk, D. Tausky, Grammar-based techniques for creating ground-truthed sketch corpora, International Journal on Document Analysis and Recognition 14 (2011) 65–74.

[16] R. Zanibbi, A. Pillay, H. Mouchère, C. Viard-Gaudin, D. Blostein, Stroke-based performance metrics for handwritten mathematical expressions, in: International Conference on Document Analysis and Recognition, 2011, pp. 334–338.

[17] R. Zanibbi, H. Mouchère, C. Viard-Gaudin, Evaluating structural pattern recognition for handwritten math via primitive label graphs, in: Document Recognition and Retrieval (DRR), 2013. doi:10.1117/12.2008409.

[18] Y. Eto, M. Suzuki, Mathematical formula recognition using virtual link network, in: International Conference on Document Analysis and Recognition, 2001, pp. 762–767.

[19] Y. Shi, H. Li, F. K. Soong, A Unified Framework for Symbol Segmentation and Recognition of Handwritten Mathematical Expressions, in: International Conference on Document Analysis and Recognition, 2007, pp. 854–858.

[20] Z. Luo, Y. Shi, F. Soong, Symbol graph based discriminative training and rescoring for improved math symbol recognition, in: IEEE International Conference on Acoustics, Speech, and Signal Processing, 2008, pp. 1953–1956.

[21] Y. Shi, F. Soong, A symbol graph based handwritten math expression recognition, in: International Conference on Pattern Recognition, 2008, pp. 1–4. doi:10.1109/ICPR.2008.4761542.

[22] K.-F. Chan, D.-Y. Yeung, Error detection, error correction and performance evaluation in on-line mathematical expression recognition, Pattern Recognition 34 (2001) 1671 – 1684.

[23] S. Lavirotte, L. Pottier, Mathematical formula recognition using graph grammar, in: Proceedings of the SPIE, Vol. 3305, 1998, pp. 44–52.

[24] R. Yamamoto, S. Sako, T. Nishimoto, S. Sagayama, On-line recognition of handwritten mathematical expressions based on stroke-based stochastic context-free grammar, IEIC Technical Report.

[25] U. Garain, B. Chaudhuri, Recognition of online handwritten mathematical expressions, IEEE Trans. on Systems, Man, and Cybernetics - Part B: Cybernetics 34 (6) (2004) 2366–2376.

[26] D. Průša, V. Hlaváč, Mathematical formulae recognition using 2d grammars, International Conference on Document Analysis and Recognition 2 (2007) 849–853.

[27] S. Lehmberg, H.-J. Winkler, M. Lang, A soft-decision approach for symbol segmentation within handwritten mathematical expressions, in: International Conference on Acoustics, Speech, and Signal Processing, Vol. 6, 1996, pp. 3434–3437.

[28] L. Hu, R. Zanibbi, Segmenting Handwritten Math Symbols Using AdaBoost and Multi-Scale Shape Context Features, in: International Conference on Document Analysis and Recognition, 2013, pp. 1180–1184.

[29] A. Thammano, S. Rugkunchon, A neural network model for online handwritten mathematical symbol recognition, in: Intelligent Computing, Vol. 4113, 2006, pp. 292–298.

[30] B. Keshari, S. Watt, Hybrid mathematical symbol recognition using support vector machines, in: International Conference on Document Analysis and Recognition, Vol. 2, 2007, pp. 859 –863.

[31] S. MacLean, G. Labahn, Elastic matching in linear time and constant space, IAPR Internatioanl Workshop on Document Analysis Systems (2010) 551–554.

[32] H.-J. Winkler, HMM-based handwritten symbol recognition using on-line and off-line features, in: IEEE Int. Conference on Acoustics, Speech, and Signal Processing, Vol. 6, 1996, pp. 3438–3441.

[33] L. Hu, R. Zanibbi, HMM-Based Recognition of Online Handwritten Mathematical Symbols Using Segmental K-Means Initialization and a Modified Pen-Up/Down Feature, in: International Conference on Document Analysis and Recognition, 2011, pp. 457–462.

[34] F. Álvaro, J. A. Sánchez, J. M. Benedí, Classification of On-line Mathematical Symbols with Hybrid Features and Recurrent Neural Networks, in: International Conference on Document Analysis and Recognition, 2013, pp. 1012–1016.

[35] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, J. Schmidhuber, A novel connectionist system for unconstrained handwriting recognition, Pattern Analysis and Machine Intelligence, IEEE Transactions on 31 (5) (2009) 855–868.

[36] M. Schuster, K. Paliwal, Bidirectional recurrent neural networks, Signal Processing, IEEE Transactions on 45 (11) (1997) 2673–2681.

[37] A. Toselli, M. Pastor, E. Vidal, On-line handwriting recognition system for tamil handwritten characters, in: Pattern Recognition and Image Analysis, Vol. 4477, 2007, pp. 370–377.

[38] F. Álvaro, J. A. Sánchez, J. M. Benedí, Offline Features for Classifying Handwritten Math Symbols with Recurrent Neural Networks, in: International Conference on Pattern Recognition, 2014, pp. 2944–2949.

[39] C. D. Manning, H. Schütze, Foundations of Statistical Natural Language Processing, The MIT Press, 1999.

[40] F. Álvaro, R. Zanibbi, A Shape-Based Layout Descriptor for Classifying Spatial Relationships in Handwritten Math, in: ACM Symposium on Document Engineering, 2013, pp. 123–126.

[41] L. Zhang, D. Blostein, R. Zanibbi, Using fuzzy logic to analyze superscript and subscript relations in handwritten mathematical expressions, in: International Conference on Document Analysis and Recognition, Vol. 2, 2005, pp. 972–976.

[42] R. Sibson, SLINK: an optimally efficient algorithm for the single-link cluster method, The Computer Journal 16 (1) (1973) 30–34.

[43] J. Goodman, Semiring parsing, Computational Linguistics 25 (4) (1999) 573–605.

[44] A. Lapointe, D. Blostein, Issues in performance evaluation: A case study of math recognition, in: International Conference on Document Analysis and Recognition, 2009, pp. 1355–1359.

[45] A.-M. Awal, H. Mouchere, C. Viard-Gaudin, The problem of handwritten mathematical expression recognition evaluation, in: International Conference on Frontiers in Handwriting Recognition, 2010, pp. 646–651.

[46] F. Álvaro, J. A. Sánchez, J. M. Benedí, Unbiased evaluation of handwritten mathematical expression recognition, in: International Conference on Frontiers in Handwriting Recognition, 2012, pp. 181–186.

[47] D. Jurafsky, J. H. Martin, Speech and Language Processing (2nd edition) (Prentice Hall Series in Artificial Intelligence), 2008.

[48] H. Ney, Stochastic Grammars and Pattern Recognition, in: Speech Recognition and Understanding, Vol. 75, 1992, pp. 319–344.

[49] J. A. Nelder, R. Mead, A simplex method for function minimization, Computer Journal 7 (1965) 308–313.

[50] M. Bisani, H. Ney, Bootstrap estimates for confidence intervals in ASR performance evaluation, in: IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 1, Montreal, 2004, pp. 409–412.