

オンライン手書き数式解答に対する深層ニューラルネットワークを用いたクラスタリングに基づく採点の研究

**A Study on Clustering-based Marking using Deep Neural Networks for Online Handwritten Mathematical Answers**

By

**UNG QUANG HUY**



東京農工大学

A thesis submitted in fulfillment  
of the requirements for the

**DEGREE OF Doctor of Philosophy OF COMPUTER SCIENCE**

At the

**TOKYO UNIVERSITY OF AGRICULTURE AND TECHNOLOGY**

Under supervision of **Prof. Masaki Nakagawa** and **Prof. Keiichi Kaneko**

© Copyright by Ung Quang Huy

Fall, 2021

## **ACKNOWLEDGMENT**

First and foremost, I would like to express my sincere gratitude to my supervisors Prof. Masaki Nakagawa and Prof. Keiichi Kaneko for the continuous support during my doctoral course and related research, with my patience, motivation, and immense knowledge. Ever since, they have supported me not only by providing me with an excellent atmosphere for doing research over three years, but also academically and emotionally through the rough road to finish this thesis. And during the most difficult times in doctoral course, they gave me the moral support and the freedom I needed to move on. In addition, I express gratitude to Prof. Richard Zanibbi for supervising me in a short-term internship at Rochester Institute of Technology, USA.

I hereby express my gratitude to KDDI Foundation, Epson Foundation and TUAT for offering me scholarships and a fellowship from April 2019 to March 2020, from April 2021 to March 2022, and from October 2021 to March 2022, respectively.

I take this opportunity to express gratitude to all of the laboratory members and iLabo Company for our valuable comments and supports, exchanges of knowledge and skills during my graduate program, especially Dr. Nguyen Tuan Cuong, Dr. Nguyen Tuan Hung, Dr. Phan Van Truyen, Dr. Phan Minh Khanh, and Dr. Vu Tran Minh Khuong.

I would also like to acknowledge professors and teachers in International Center of Tokyo University of Agriculture and Technology for teaching me Japanese language since the first days I came to Japan. Thank this support, now I can communicate with Japanese people and enjoy life in Japan.

I also thank my parents and my friends in Vietnam for unceasing encouragement, support and attention.

## ABSTRACT

This thesis presents a study on clustering-based marking using deep neural networks for online handwritten mathematical answers. In education, descriptive math questions are considered to be better than multiple-choice math questions to evaluate students' understanding and abilities to answer. However, examiners need to scan and score a large number of answers, which requires a huge amount of time and effort. Clustering-based marking, a quite new topic in the research community, is a promising approach to help examiners to mark handwritten answers. In this thesis, we present three main contributions: (1) we present our strategy and two tools (e-testing tool and e-marking tool) for collecting and annotating handwritten descriptive answers, (2) we present two approaches for clustering online handwritten mathematical expressions (OHMEs), (3) we present two methods for improving OHME recognition.

First, we propose an e-testing tool on a tablet, which works as the pattern collection tool, and an e-marking tool as the annotation tool for creating a dataset of handwritten math answers (HMAs) for descriptive questions. We present specifications and workflows of those tools in detail. By providing the e-testing tool and the e-marking tool, we plan to collaborate with other organizations for collecting a large dataset, then publish it for the research community.

Secondly, we propose two approaches for clustering OHMEs to create a clustering-based marking. To the best of our knowledge, we are the first group attempting to cluster OHMEs. Mathematical expressions are 2D-structural and infinite combinations of math symbols and spatial relationships. Our first approach is to extract features from low-level pattern features to high-level symbolic and structural features obtained from processing and recognizing OHMEs. The second approach is to compute pairwise similarities among OHMEs. We achieved the best results of around 0.916 and 0.915 for purity and around 0.556 and 0.702 for the marking cost on two answer datasets, Dset\_Mix and NIER\_CBT, respectively.

Thirdly, we propose two methods for improving OHME recognition. Since our proposed clustering methods utilize the recognition results of OHMEs, we aim to improve the recognition rate for improving the performance of the clustering process. The first method is to utilize bidirectional context from input stroke sequences for symbol segmentation and classification. The second method is to utilize a math language model combined with OHME recognizers. We propose the first transformer-based math language model which can combine with both online and offline HME recognizers. Experiments showed that our proposed methods can improve the performance of OHME recognizers.

# LIST OF CONTENTS

ACKNOWLEDGMENT .....	2
ABSTRACT .....	3
LIST OF FIGURES .....	6
LIST OF TABLES .....	8
LIST OF ABBREVIATIONS AND TERMINOLOGIES .....	9
CHAPTER 1. Introduction .....	10
1.1. Backgrounds .....	10
1.2. Contributions .....	11
1.3. Thesis organization .....	12
CHAPTER 2. Surveys .....	13
2.1. Surveys on HME collecting tools .....	13
2.2. HME clustering .....	13
2.3. Computer-assisted marking .....	14
2.4. Online HME recognition .....	14
2.4.1. A general framework of OHME recognizer .....	14
2.4.2. OHME recognition methods .....	16
2.4.3. Language models for HME recognition .....	18
CHAPTER 3. Strategy and tools for collecting and annotating handwritten descriptive answers for developing automatic and semi-automatic marking - an initial effort to math     21	
3.1. Introduction .....	21
3.2. Toward Handwritten Exam Answer Database .....	21
3.3. Specifications of the E-testing Tool .....	22
3.4. Specifications of the E-marking Tool .....	23
3.5. HMA Collection Process .....	24
3.5.1. Question Preparation .....	25
3.5.2. Collection Process .....	25
3.5.3. Design Considerations .....	26

3.6.	HMA Annotation Process .....	27
3.6.1.	Annotation Process .....	27
3.6.2.	Output Structure and Format .....	29
CHAPTER 4.	Clustering online handwritten mathematical answers.....	31
4.1.	Introduction.....	31
4.2.	Our proposed methods .....	31
4.2.1.	Multi-level features from OHMEs .....	32
4.2.2.	Distance-based representation .....	36
4.2.3.	Generative sequence similarity function based on a Seq2Seq model .....	37
4.3.	Measurements for clustering-based marking .....	40
4.4.	Experiments .....	41
4.4.1.	Experiments on multi-level features of OHMEs .....	41
4.4.2.	Experiments on generative sequence similarity function .....	49
4.5.	Conclusions.....	55
CHAPTER 5.	Online Handwritten Mathematical Expression Recognition.....	57
5.1.	Introduction.....	57
5.2.	Proposed methods .....	58
5.2.1.	Online handwritten mathematical symbol segmentation and recognition with bidirectional context.....	58
5.2.2.	Transformer-based math language model .....	62
5.3.	Experiments .....	65
5.3.1.	Experiments on online handwritten mathematical symbol segmentation and recognition.....	65
5.3.2.	Experiments on transformer-based math language model .....	69
5.4.	Conclusions.....	75
CHAPTER 6.	Conclusion and future works.....	76
6.1.	Conclusions.....	76
6.2.	Future works .....	76
REFERENCES	.....	77
APENDIX I – PUBLICATIONS	.....	83

## LIST OF FIGURES

Figure 1.1. Overview of computer-assisted marking .....	11
Figure 2.1. Flow of approaches applied for recognizing OHMEs.....	15
Figure 2.2. Example of grammar rules and its parsing tree. Note that each rule in the grammar takes one of two forms: $X \rightarrow Y_1 Y_2$ where $X \in N$ , $Y_1 \in N$ , $Y_2 \in N$ ; or $X \rightarrow Y$ where $X \in N$ , $Y \in \Sigma$ .....	17
Figure 3.1. An example of an XML format for 2 Math questions. ....	25
Figure 3.2. Main interface of our e-testing tool.....	26
Figure 3.3. Main interface for marking multiple HMAs. ....	28
Figure 3.4. Main interface for marking a single HMA.....	29
Figure 4.1. Types of features. ....	32
Figure 4.2. Three main steps for extracting directional features. ....	33
Figure 4.3. Bag-of-symbols and bag-of-relations for a given OHME, where “recog.” and “hor” are abbreviations of “recognition” and “horizontal” respectively. ....	34
Figure 4.4. Example for dividing SRT consisting of the six types of spatial relationships.....	35
Figure 4.5. Illustration of using the largest SRT to divide others. ....	35
Figure 4.6. Illustration of dividing an SRT into $M \times N$ positions, performing zero padding, and applying a Gaussian filter over the position and its neighbors. ....	36
Figure 4.7. Clustering process with pairwise similarity function.....	37
Figure 4.8. A standard Seq2Seq model. ....	38
Figure 4.9. Illustration of computing $F(S_1 S_2)$ .....	39
Figure 4.10. Samples in subgroup 6 of Dset_Mix.....	42
Figure 4.11. Average (lines) and standard deviation (light color areas) of purity, those of the marking cost, and those of #OHMEs in each cluster for Dset_Mix with feature combination in E10 for increasing number of clusters.....	47
Figure 4.12. Overview of TAP consisting of point-based features as the input (A), the encoder part (B), the decoder part (C), and the output (D).....	50
Figure 4.13. Details of NIER_CBT. (a) shows the number of correct\incorrect categories in each question, and (b) shows the number of correct\incorrect patterns in each question.....	50

Figure 4.14. Visualization of the SbR matrix of the subgroup 3 and 8 before normalizing them into [0, 1]. .....	55
Figure 5.1. Symbol classification by strokes query .....	58
Figure 5.2. Temporal classification probability: (a) single symbol (b) two symbols. Dashed line shows the probability of ‘blank’ symbol. ....	60
Figure 5.3. Overview of the proposed transformer-based language model with two transformer layers. ....	63
Figure 5.4. Illustration of scale dot-product attention and masked multi-head attention. ....	64
Figure 5.5. HME recognition time according to the number of strokes. ....	69
Figure 5.6. Illustration for symbol-relation temporal classifier.....	70
Figure 5.7. Symbol-level parser. ....	70
Figure 5.8. Examples of corrected and miscorrected cases when combining the SRTC_SLP recognizer and TMLM_8L (LM: language model). Each case shows an HME image, its ground truth, and its recognition candidates with/without TMLM_8L and their corresponding scores from TMLM_8L. .	74

## LIST OF TABLES

Table 3.1. A Description of available functions in rectangle 5 of Figure 3.2. ....	27
Table 3.2. A description of available functions in Figure 3.3 and Figure 3.4 .....	29
Table 4.1. Details of the Dset_50 dataset. ....	41
Table 4.2. Details of the Dset_Mix dataset. ....	42
Table 4.3. Expression recognition rate and F1-score of symbol and spatial relationship recognition. ....	44
Table 4.4. Experiment settings on single types of features and their combinations	45
Table 4.5. Experiments on single types of features and their combinations for Dset_50 and Dset_Mix. ....	45
Table 4.6. Results of $k$ estimation and marking cost ( $MC$ ). ....	48
Table 4.7. Comparison with other research on Dset_50 and Dset_Mix.....	48
Table 4.8. ExpRate and CER of MTAP. ....	52
Table 4.9. Comparisons with other methods of clustering HMEs. Values are presented in form of “average value (standard deviation)” .....	53
Table 4.10. Comparisons with other variants of SFs.....	54
Table 5.1. CROHME 2016 dataset.....	65
Table 5.2. Symbol classification in CROHME 2016. ....	67
Table 5.3. Class-based recognition comparison. ....	67
Table 5.4. Expression rate (%) compare with state-of-the-arts on CROHME 2016. ....	68
Table 5.5. Comparisons with other language modeling methods.....	72
Table 5.6. Expression rates on combining the HME recognizers with language models.....	73
Table 5.7. Percentages of corrected, miscorrected, and unchanged recognition results when combining the SRTC_SLP recognizer with language models.....	73



## LIST OF ABBREVIATIONS AND TERMINOLOGIES

ME	Mathematical Expression
HME	Handwritten Mathematical Expression
HMA	Handwritten Mathematical Answer
OHME	Online Handwritten Mathematical Expression
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long-Short Term Memory
GRU	Gated Recurrent Unit
SCFG	Stochastic Context-Free Grammars
CYK	Cocke-Younger-Kasami
MST	Maximum Spanning Tree
LOS	Line-of-Sight
SRT	Symbol Relation Tree
BoS	Bag-of-symbols
BoR	Bag-of-relations
BoP	Bag-of-positions
PbBoS	Position-Based Bag-of-Symbols
PbBoR	Position-Based Bag-of-Relations
CROHME	Competition on Recognition of Online Handwritten Mathematical Expressions
DbR	Distance-based Representation
SbR	Similarity-based Representation
GSSF	Generative Sequence Similarity Function
TMLM	Transformer-based Math Language Model

# CHAPTER 1. Introduction

## 1.1. Backgrounds

In 2020, the widespread of the SARS-CoV-2 (COVID-19) has a strong impact on education over the world, which caused almost schools and universities to be temporally closed. Many educational organizers resume the learners' studies via online platforms in response to significant demand. The adoption of online learning might continue persisting in post-pandemic, and the shift would impact the worldwide education market. In this context, the self-learning and e-testing applications would be necessary options for students in the near future.

Nowadays, touch-based and pen-based devices are becoming very popular as learning media. Children and students use them to read textbooks and exercise. Moreover, they are suitable for learners to write mathematical expressions (MEs), which could be better than using editors such as Microsoft Equation Editor, MathType, or high-quality typesetting systems like LaTeX.

Over the past two decades, research on how to better recognize handwritten mathematical expressions (HMEs) has significantly increased due to increased demands for its application on tablets. Competitions on recognizing online HMEs (OHMEs) have been ongoing under the series of CROHME [1] with improved recognition performance. With this progress, many e-learning interfaces based on pen-based devices have been studied [2–4] and employed in practical applications. If the recognition result is verified and confirmed by a learner, either online or offline HME recognition can be incorporated into self-learning and e-testing applications. Although a learner has to do additional work, the learner can get immediate feedback.

HME recognition can also be used for marking. Automatic marking by comparing the recognition result of an HME answer with the correct answer is one of the solutions for marking many answers. However, there remains problems [5]. Firstly, it is not so simple to mark partially correct answers. Secondly, there may be several correct answers as well as some different but equivalent notations for an answer. It is hard to pre-define all possible cases. Thirdly, it requires examiners or examinees to confirm the automatic marking since the recognition result is not always correct. In fact, examinees in large-scale examinations (e.g., national-wide qualification examinations) do not have opportunities to confirm the marking so that examiners or someone else must confirm the marking.

Computer-assisted marking as shown in Figure 1.1 is an alternative approach. One of the most promising applications of computer assistance is clustering answers, which groups similar answers that could be given the same score. If answers are well clustered, they

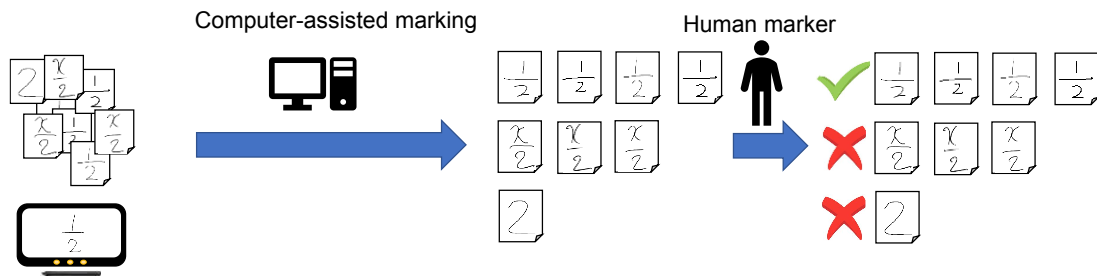


Figure 1.1. Overview of computer-assisted marking

can be marked efficiently, and marking errors can be decreased. Since the final marking is made by human markers, examinees' anxieties will also decrease.

To make computer-assisted exam marking useful, however, several problems need to be solved. Firstly, HME recognition is one of the most difficult handwriting recognitions. We must be able to cluster them even though their recognition is fragile. Secondly, there are several clustering algorithms using different approaches. We need to choose a clustering mechanism suitable for the nature of the data such as the number of clusters and the distribution of the data. In this thesis, we focus on clustering and recognition of OHME to make a clustering-based marking.

## 1.2. Contributions

First, we propose an e-testing tool on a tablet, which works as the pattern collection tool and an e-marking tool as the annotation tool for creating a dataset of handwritten math answers (HMAs) for descriptive questions. As my best of knowledge, there is currently no published large dataset of handwritten math answers for descriptive questions. In our laboratory, we generated and published a synthetic dataset for evaluating our proposed method temporally. An answer dataset is collected by a collaboration with National Institute for Educational Policy Research (NIER) in Tokyo, Japan. However, this dataset is small, and it just contains a small portion of common math symbols. By providing the e-testing tool and the e-marking tool, we plan to collaborate with other organizations for collecting large dataset, then publishing for the research community.

Secondly, we propose two approaches for clustering OHMEs to create a clustering-based marking. As my best of knowledge, we are the first group attempting to cluster OHMEs. MEs are 2D-structural and infinite combinations of math symbols and spatial relationships. Our first approach is to extract features from low-level pattern features to high-level symbolic and structural features obtained from processing and recognizing OHMEs. The extracted features are then transformed to a distance-based representation and inputted to a clustering algorithm for producing groups of OHMEs. The second approach is to compute pairwise similarities among OHMEs. Experiments showed the effectiveness of our methods in term of reducing the marking cost.

Thirdly, we propose two methods for improving OHME recognition. Since our proposed clustering methods utilize the recognition results of OHMEs, we aim to improve the recognition rate for improving the performance of the clustering process. The first method is to utilize bidirectional context from input stroke sequences for symbol segmentation and classification. The second method is to utilize a math language model combined with OHME recognizers. We propose the first transformer-based math language model which can combine with both online and offline HME recognizers. Experiments showed that our proposed methods can improve the performance of OHME recognizers.

### **1.3. Thesis organization**

Chapter 2 presents the surveys on three above topics. Chapter 3 presents a strategy and tools for collecting and annotating HMAs. Chapter 4 presents our proposed methods for clustering OHMEs. Chapter 5 presents our proposed methods for improving the recognition rate of OHME recognition. Finally, chapter 6 gives some conclusions and future works.

## CHAPTER 2. Surveys

### 2.1. Surveys on HME collecting tools

In the recent decade, sample patterns for handwritten mathematical expressions (HMEs) have been collected and made available for the HME recognition community, since HME recognition has been studied intensively as the most natural method for inputting MEs rather than math editors such as Microsoft Equation Editor and math description languages such as LaTeX [6,7]. Several projects have been conducted for collecting online HME patterns by MathBrush [8], Awal et al. [9], Quiniou et al. [10], Stria et al. [11], Aguilar et al. [12], etc. The collected HME patterns have been gathered to form a large sample dataset for the competitions on recognition of HMEs (CROHME) [6]. Some of the products developed along the competitions are commercially available in the market such as the products by MyScript and Wiris. Several tablet-based e-learning interfaces have been researched [2–4] and employed for practical applications.

Most tools to collect HME patterns use the same process for capturing online HME patterns. Pen movements captured from a tablet are stored as a list of successive pen-tip points with each element showing x and y coordinates of the pen-tip at a time step. In mathematics, each symbol is a group of one or more strokes where a stroke is a sequence of pen-tip points from pen-down to pen-up. The patterns thus captured are called online patterns, while images captured from a camera is called offline patterns. Offline patterns are easily converted from online patterns by rendering them to bitmap images. These tools are made for internal use and have not been released publicly.

### 2.2. HME clustering

There are several past works on clustering offline (bitmap image) HMEs. Khuong et al. [13] combined low-level features (directional features) and high-levels features (bag-of-symbols, bag-of-relations, and bag-of-positions) to represent each offline HME. However, the high-level features are formed from offline isolated symbols classified from connected components along with predefined heuristic rules. Hence, there still exist problems related to segmentation and determination of spatial relationships. Recently, Nguyen et al. [14] presented features based on spatial classification using a convolutional neural network (CNN). Their model is trained to localize and classify objects (symbols) in each offline HME via weakly supervised learning. Then, spatial pooling is applied to extract hierarchical spatial classification features from the class activation maps.

In this work, we aim to address the problem of extracting features from sequential data for clustering OHMEs. A similar problem exists on natural language processing, where the vector space needs to be constructed to represent a word sequence  $\{w_1, w_2, \dots, w_n\}$  and express the corresponding semantics. A conventional method to address this problem

is via bag-of-words [15]. However, a drawback of this method is that the word order is lost; hence, uniqueness is not guaranteed. A better solution proposed by Le and Mikolov[16] is to construct a continuous vector space, where semantically similar sequences are mapped onto nearby points. Besides, these vectors can be applied to machine learning algorithms such as k-means and support vector machines.

There are deep neural network-based methods for clustering sequential data such as OHMEs. Several methods aim to embed sequential data into feature vectors based on the reconstruction loss and the clustering loss [17,18]. Another approach is to compute the pairwise similarity/dissimilarity instead of embedded features [19]. However, those methods without information about symbols and relations encounter difficulty for clustering OHMEs since there are infinite combinations of symbols and relations to form MEs. Nguyen et al. [14] showed that metric learning methods do not work well compared to CNN-based spatial classification features for clustering offline HMEs.

### **2.3. Computer-assisted marking**

Extensive research has been carried out on essay assessment [20–22] and handwritten essay scoring [23]. Basu et al. proposed a method for clustering answers for English short answer grading [24]. They trained a similarity metric to calculate a distance between two different answers using logistic regression. Then, they employed a modified k-Medoids and a latent Dirichlet allocation algorithm for forming clusters and sub-clusters of answers. The clustering method allows graders to just score each group using one operation, which reduces the cost of the grading process. Brooks et al. used the approach to design a cluster-based interface [25], which is effective because it allows graders to give feedback for clusters and sub-clusters of answers at once.

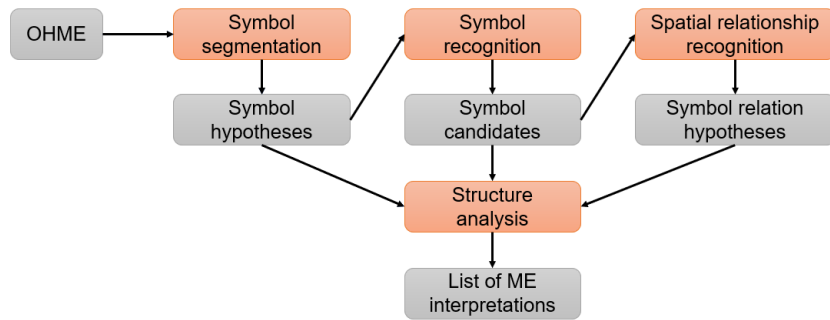
To assess handwritten paper-based work, Singh et al. [26] introduced a web-based system that allows students to upload their scanned assignments. The system also allows teachers to categorize the answers and give feedback for them. User reports from four years of usage of the system demonstrate its effectiveness in terms of speed, consistency, and flexibility.

### **2.4. Online HME recognition**

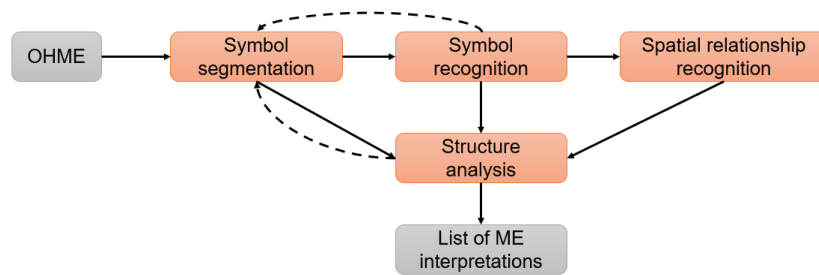
This section presents a common framework for the OHME recognizer and current several approaches to solve this problem.

#### **2.4.1. A general framework of OHME recognizer**

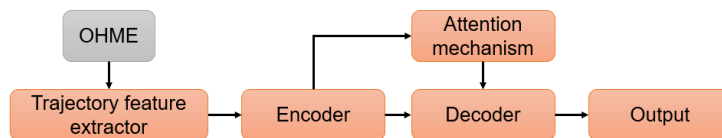
Recognition of HMEs is one of the current challenges in handwriting recognition. It consists of four main tasks: symbol segmentation, symbol recognition, spatial relationship classification, and ME structure analysis. Figure 2.1 shows common architectures for OHME recognition with its key modules.



(a) A flow of the general approach with four major modules



(b) A flow of the approach having modular driven by other module(s). The dashed arrows indicate the driving modules and driven modules.



(c) A flow of the end-to-end approach.

Figure 2.1. Flow of approaches applied for recognizing OHMEs.

Given an OHME, symbol segmentation firstly produces symbol hypotheses - each of which may form a symbol; the symbol recognition secondly proposes a list of symbols for each symbol hypothesis; spatial relationship classification thirdly identifies the relation between two symbol hypotheses and forms a symbol relation tree of an OHME, and structure analysis finally determines the best ME interpretations. In many approaches, tasks could be processed sequentially, or an earlier task can be processed in parallel with a later process. For example, symbol segmentation could be joined with symbol recognition, and relation classification could be integrated into structure analysis and so on.

Several systems perform more tasks, such as pre-processing, normalization, noise reduction, training language models, and so on. For pre-processing OHMEs, it usually

utilizes resampling and smoothing [27]. Normalization and interpolation are often applied to make the later processing easier such as writing speed normalization and size normalization [28]. Preparing or constructing a language model for ME is crucial to assist structure analysis, but it is considered a sub-task of the structure analysis.

The end-to-end approach has been proposed for recognizing OHME, which commonly includes four main components: trajectory feature extractor, encoder, decoder, and attention mechanism, as shown in Figure 2.1(c). Although it utilizes layers of neural networks to recognize input OHMEs, they still have functions corresponding to those key tasks. Particularly, the encoder firstly exploits useful features and represent them as high-level features, symbol segmentation is carried out via the attention mechanism; secondly, the decoder the focused parts in encoded features, which pointed out by the attention model; spatial relationship classification is also made based on the attention model that guides the decoder precisely attend to the direction between the current predicted symbol and the next predicted symbol; the combination of this information forms a 2D structure of a ME often represented by a LaTeX string. Note that the end-to-end recognizer does not need to pre-defined grammar rules of ME so that they could output invalid structure of LaTeX strings. To address this problem, a pre-trained or simultaneous-trained language model is jointly applied to learn the long-term dependencies of math symbols.

## **2.4.2. OHME recognition methods**

In this section, we present three common approaches for OHME recognition.

### **2.4.2.1. ME grammar-based methods**

Same as any language, mathematics can be described by the grammar. Generation of a complete ME requires not only a correct combination of symbol segmentation hypotheses, symbol recognition candidates, and spatial relationships but also correct mathematical syntaxes. Many studies have been conducted on ME grammar-based methods, which utilize 2-dimensional stochastic context-free grammars (2D-SCFG) with Cocke-Younger-Kasami (CYK) algorithm such as [28–33]. We briefly describe these two techniques here.

In formal language theory, a context-free grammar (CFG) contains a set of derivation rules with no priorities. A sentence may have multiple parses in a CFG, and all these parses are equivalent. The probabilities of occurrences of these parses in mathematics are not the same. Hence, to assign probability factors to multiple parses of a sentence, Stochastic context-free grammars (SCFG) are presented, which have an additional property that each grammar rule is defined with a score that can be set manually or obtained through a training procedure. Then, 2D-SCFG is defined by adding a finite set of relations between two elements to represent 2-dimensional languages like mathematics. Figure 2.2 illustrates the 2D-SCFG by presenting an example of the grammar rules and a parsing tree of an ME “ $x_m + 1$ ”. To parse a tree from the given set of grammar rules, we



Let  $\rightarrow \{a, b, c, d, e, f\}$

Num  $\rightarrow \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

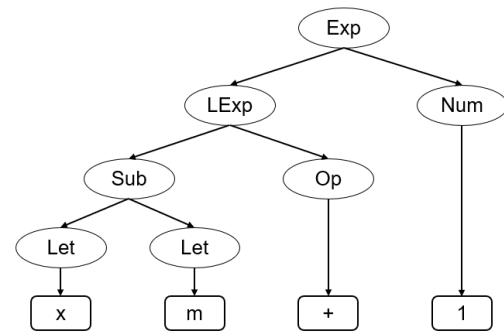
Op  $\rightarrow \{+, -, \backslash\text{times}, \backslash\text{div}\}$

Sub  $\xrightarrow{\text{subscript}} \text{Let Let}$

LExp  $\xrightarrow{\text{horizontal}} \text{Sub Op}$

Exp  $\xrightarrow{\text{horizontal}} \text{LExp Num}$

(a) A set of grammar rules



(b) A parsing tree of 2D-SCFG for “ $x_m + 1$ ”.

Figure 2.2. Example of grammar rules and its parsing tree. Note that each rule in the grammar takes one of two forms:  $X \rightarrow Y_1 Y_2$  where  $X \in N$ ,  $Y_1 \in N$ ,  $Y_2 \in N$ ; or  $X \rightarrow Y$  where  $X \in N$ ,  $Y \in \Sigma$

extract the symbols from “ $x_m + 1$ ” where “ $x$ ” is a letter (denoted by “Let”), “1” are numbers (denoted by “Num”), and “+” is an operator (denoted by “Op”). Then, new non-terminal nodes are formed from one or two non-terminal nodes according to the unary or binary grammar rules, respectively. This step is repeated until we get a non-terminal node that cannot produce any non-terminal node.

Cocke-Younger-Kasami (CYK) algorithm, a dynamic programming algorithm, is used to parse an HME with 2D-SCFG. It is a bottom-up parser that constructs a CYK table of terminal and non-terminal symbols by combining smaller substrings into larger substrings in a bottom-up scheme. A stochastic version of the CYK algorithm computes the probabilities of possible derivations of a given string. The complexity of CYK is  $O(n^3|G|)$ , where  $n$  is the length of the parsed string and  $|G|$  is the size of the grammar set  $G$ . A recognition score is a combination of the segmentation score, recognition score, relation score, and grammar score. The first candidate in the top cell is the final recognition result of the input HME.

#### 2.4.2.2. ME graph-based methods

In graph-based methods, symbol recognition and spatial relationship recognition of an OHME are represented as a graph, in which each node represents a symbol hypothesis, and each edge represents the relationship between two symbol hypotheses. Each path or sub-graph follows some constraints and generates a possible interpretation for a sub-expression within a given OHME. All their possible interpretations are evaluated by combining segmentation scores and classification scores, and the interpretation with the best score is selected as the recognition result. Since MEs follow a hierarchical language, ME grammar-based methods and ME graph-based methods often use appearance or

content-based encoding to represent an ME in the form of a symbol relation (or layout) tree or an operator tree.

There are many publications on graph-based methods. The early graph-based methods for OHME recognition [34–36] are based on probability maximization or penalty minimization. These methods share a common approach in that all possible symbol hypotheses are computed from the symbol segmentation and the symbol recognition scores. An advantage of these methods is that local recognition errors can be corrected by considering the evaluation of the full recognition tree. In these methods, the problem of OHME recognition is transformed into finding the most probable sequence of symbols and their relations within the generated graph. Hu et al. [37] proposed a parser based on Minimum Spanning Tree (MST) with a language model embedded in the set of classes and MST extraction algorithm as a graph-based parsing method without using grammars. First, they constructed a directed Line-of-Sight (LOS) graph in which an edge is a line of sight from the center of a stroke to the convex hull of every other stroke. An MST, which represents a symbol layout, is formed by applying the Edmonds’ algorithm on LOS graph. Among graph-based methods, some methods employ graph grammars to resolve ambiguities in an ME graph or a sub-graph, which are called graph grammar-based methods in [38,39].

#### **2.4.2.3. End-to-end methods**

Seq2Seq models for recognizing OHMEs are deep learning models that directly convert an input sequence into an output sequence. A Seq2Seq model consists of two main components, an encoder and a decoder. The encoder, an LSTM or a BLSTM network, accepts a time series input of arbitrary length and encodes information from the input into a hidden state vector. The decoder, commonly an LSTM network, generates a sequence corresponding to the input sequence.

Zhang et al. [27] proposed a track, attend and parse (TAP) architecture, which parses an OHME into a LaTeX sequence by tracking a sequence of input points. The encoder or the tracker stacks several layers of bidirectional Gated Recurrent Units (GRUs) to get the high-level representation. The decoder or the parser is composed of unidirectional GRUs combined with a hybrid attention mechanism and a GRU-based language model. The hybrid attention consists of two attentions: coverage based spatial attention and temporal attention. Hong et al. [40] improved TAP by adding residual connections in the encoder to strengthen the feature extraction and jointly using a transition probability matrix in the decoder to learn the long-term dependencies of mathematical symbols.

#### **2.4.3. Language models for HME recognition**

There remain challenging problems in HME recognition. One problem is that there are lots of ambiguities in the interpretation of HMEs. For instance, there exist math symbols that are very similar in the writing style, such as “0”, “o”, and “O” or dot and comma.

These ambiguities challenge HME recognition without utilizing contextual information. In addition, recognition systems without using predefined grammar rules such as the encoder-decoder model [27,41] might result in syntactically unacceptable misrecognitions. One promising solution for these problems is to combine an HME recognition system with a math language model. Employing language models for handwritten text recognition has shown effectiveness in previous research [42–44].

An ME has a 2D structure represented by several formats such as MathML, one-dimensional LaTeX sequences, and two-dimensional symbol layout trees [7]. Almost all recent HME recognition systems output their predictions as the LaTeX sequences since LaTeX is commonly used in real applications.

There are some common challenges in modeling MEs similar to natural language processing. First, there is a lack of corpora of MEs as MEs rarely appear in daily documents. Secondly, there are infinite combinations of symbols and spatial relationships in MEs. Thirdly, there are long-term dependencies and correlations among symbols and relations in an ME. For example, “(” and “)” are often used to contain a sub-expression, and if they contain a long sub-expression, it is challenging to learn the dependency between them.

There are several methods to modeling MEs. The statistical  $N$ -gram model was used in [45]. It assigns a probability for the  $n$ -th tokens given  $(n-1)$  previous tokens based on the maximum likelihood estimation. However, the  $N$ -gram model might not represent the long dependencies due to the limitation of the context length. Increasing this length might lead to the problem of estimating a high-dimensional distribution, and it requires a sufficient amount of training corpus. In practical applications, the trigram model is usually used, and the 5-gram model is more effective when the training data is sufficient. The recurrent neural network-based language model (RNNLM) proposed by [46] was utilized in HME recognition systems [1,27]. RNNLM predicts the  $n$ -th token given  $(n-1)$  previous tokens in previous time steps. However, they still face the problem of the long-term dependencies.

Language models are well-known as generative models and autoregressive models since they predict the next state of a variable given its previous states. In NLP, Radford et al. [47,48] proposed Generative Pre-Training models (GPT and GPT-2) with high achievements on NLP benchmarks based on the vanilla transformer-based network in [49]. Their models are trained by the casual language modeling loss, then fine-tuned for multitask learning such as text classification, question answering, and similarity. Dai et al. [50] presented a Transformer-XL for capturing extended length of context using a recurrent architecture for context segments. Transformer-XL can learn dependency that is 80% longer than RNNs, 450% longer than TLM. The inference speed is 1,800 times faster than TLM by caching and reusing previous computations. XLNet presented by Yang et al. [51], is the first model utilizing bidirectional contexts for transformer-based

language models. This model significantly outperformed the conventional BERT model [52] in 20 tasks of NLP benchmarks.

There are several studies combining HME recognition systems with pre-trained language models. Wu et al. [45] combined their encoder-decoder HME recognizer with a pre-trained 4-gram model to get the  $N$  best paths. Zhang et al. [27] utilized a Gated Recurrent Unit-based language model (GRULM) for their HME recognizer that is an encoder-decoder model with temporal attention. This attention is to help the decoder determine the reliability of spatial attention and that of the language model per time step. The language models improved the expression rate by around 1 percentage point. Hence, the approach for combining language models into recognition systems is essential to study.

In CROHME 2019 [1], the Samsung R&D team used a probabilistic context-free grammar-based recognizer combined with two bigram language models, i.e., a language sequence model and a language model for spatial relationships. Besides, the MyScript team used LSTM-based language models for their grammar-based recognition system.

## **CHAPTER 3. Strategy and tools for collecting and annotating handwritten descriptive answers for developing automatic and semi-automatic marking - an initial effort to math**

### **3.1. Introduction**

Descriptive questions can far better test learners' understanding and abilities to think than multiple-choice questions for which learners can select correct answers by chance. Moreover, descriptive questions foster learners to think rather than to select. On the other hand, the drawback of descriptive questions is that it requires large time and effort for marking them. Therefore, automatic and semi-automatic marking is sought recently.

For both automatic and semi-automatic marking, we need a large database of handwritten answers for descriptive questions to support research and development on automatic and semi-automatic marking. As the database is larger, the effect and reliability of the research are more significant. This database will also be useful for handwriting recognition research since it is a collection of most casually and naturally written patterns.

We can employ the tablet-based exam since a tablet is the best device where an examinee can input his/her profile, read a question and write an answer. Moreover, online trajectories can be easily converted to offline so that online and offline recognition and clustering methods can be tested.

To make a database of handwritten answers, we propose an e-testing tool on a tablet, which works as the pattern collection tool and an e-marking tool as the annotation tool. Those tools are useful to collect natural and casual handwritten answers, although annotating ground-truth is challenging.

Here, we focus on handwritten math answers, since math is the subject for which descriptive questions are most effective. Moreover, the performance of recognizing handwritten math expressions is still lower than natural languages so that more sample patterns, which are casually and naturally written, are needed to improve their recognition.

This chapter is organized as follows. Section 3.2 presents our strategy to build an exam answer database, not restricted to math answers. Section 3.3 and section 3.4 describes the specification of e-testing tool and e-marking tool, respectively. Section 3.5 and section 3.6 describes processes for an HMA collection and annotation, respectively. The HMAs collection methodology and related issues are described in section 3.6.

### **3.2. Toward Handwritten Exam Answer Database**

One way to collect handwritten patterns is to ask each participant to write according to predefined ground-truth. Although we design the collection methods to collect natural patterns, in such a way that a meaningful sentence is shown and it is written in the flow of the sentence, it is basically “copy style”. This style is used to collect handwriting databases such as online handwritten Japanese text [53] and Vietnamese [54]. The merit of this style is that a required vocabulary can be covered and the labor to provide ground-truth is lightened. Erroneous patterns for ground-truth may be deleted. However, collected patterns may not be completely natural. The other way to overcome this problem is to let participants write whatever they want to write. Later, ground-truth is provided manually using some tools. This collection is called “freestyle” by Matsushita et al. in [55]. The drawback of this style is that it requires a large time and effort to provide ground-truth for collected handwritten samples. To make research on automatic and semi-automatic marking, however, the freestyle to collect handwritten answer patterns must be selected.

Therefore, we have set up a strategy to build a large database of handwritten exam answers. We first provide two tools:

1. A simple e-testing tool, which accepts a list of questions from an examiner, display each question on a tablet and allows each examinee to write an answer for each question. This E-testing tool should collect the profile of the examiner when he/she registers a list of questions and the profiles of all the examinees when they answer the questions.
2. An e-marking tool, by which the simplest ground-truth of “correct” or “incorrect” is tagged to each handwritten answer. Intermediate scores for partially correct answers should also be given. More detailed ground-truth can be provided by editing and annotating functions to segment lines into sentences, words, characters, mathematical expressions, etc., and provide ground-truth to each object. Recognition engines can be incorporated.

Since the work for preparing questions, answering questions, and marking answers require large time and effort, we ask collaborations who want to share the large data set of annotated handwritten answers. We also invite volunteers.

This way of collecting handwritten answer patterns enables us to collect natural and casual handwriting rather than those copied or simulated. Although it is labor-intensive, once it is made, it is also useful for developing handwriting recognition methods.

We provide these tools rather than define the common format for an exchange since we expect many institutions to collaborate to make a large database of handwritten answer patterns.

### **3.3. Specifications of the E-testing Tool**

We propose the following requirements for designing the e-testing tool:

- R1: The profile of each participant must be recorded, but some privacy information must be concealed. This information is useful for further research, such as analysis and improvement of user experience.
- R2: Multiple lines of an HMA should be allowed since people may need to write many intermediate steps before getting the final result.
- R3: A friendly user interface that navigates an answerer from a question to another or goes back to any question to revise the answer easily.
- R4: The user interface should use common and meaningful icons or images to replace or minimize the display of explanatory text since the users may come from many cultures and countries. It must support the methods for undo/redo, erasing and rewriting HMAs.
- R5: Multiple types of pen-based devices should be allowed. Then, the specification of each device used must be recorded, such as its type, sampling rate, spatial resolution, and additional information such as pressure. Among various pen input devices on the market, we prefer to use PCs or hybrid tablets or external digitizing tablets (e.g., from Wacom) connected to PCs.
- R6: The sampling rate and resolution should be high enough to capture quick pen movement since the pen trace is usually sampled with a constant rate and thus pen-tip points are evenly in time but not in space.
- R7: The output format must follow the common format being used by the community such as Ink Markup Language (InkML) format [56] for online patterns and bitmap images for offline patterns.

For intermediate steps of answers, we consider using them in further research in the future. Here, our target is to build general tools for collecting patterns as much as possible for future research.

### **3.4. Specifications of the E-marking Tool**

This section presents the basic requirements for the e-marking tool:

- R1: The user interface must show each question, participants' answers, and the specimen answers.
- R2: It should provide a menu (checkbox, up/down counter, slider, etc.) for a human marker to select correct, wrong, or partial points.
- R3: The interface should allow the human marker to utilize two common strategies in assessment: marking all answers of each person before moving to

other ones, and marking all answers to the same question in a group or the whole of the participants. The first way allows the marker to completely tag the ground-truths for the answers by a single participant while the second one helps them to concentrate on assessing the answers for each question, avoiding making mistakes or variations among participants. Moreover, the second method is possible to apply a computer-assisted marking, such as a clustering-based approach [5,57] in order to reduce the marking cost.

- R4: This user interface, again the same as the e-testing tool, should use common and meaningful icons or images to replace or minimize the display of explanatory text.
- R5: Showing the current status of marking progress, as well as marking results (annotation tags) might be useful. This information is useful to support the marker to mark, review, and revise marking. In addition, this display allows the marker to be able to utilize suitable marking strategies and make effective collaborations among human markers.
- R6: Sort and search functions might be useful for the marker to search particular items for reviews and revisions.

### **3.5. HMA Collection Process**

This section describes question preparation, process to collect HMAs and some design considerations.



```

<question>
  <annotationXML>
    <numQuestion>2</numQuestion>
    <timeLimit>10M</timeLimit>
    <question_1>
      <point>2</point>
      <content>
        Solve  $x$ :
        
$$3x^2+5x+3=0$$

      </content>
    </question_1>
    <question_2>
      <point>2</point>
      <content>
        Simplify the following expression:
        
$$\frac{1}{2} + \frac{1}{3}$$

      </content>
    </question_2>
  </annotationXML>
</question>

```

Figure 3.1. An example of an XML format for 2 Math questions.

### 3.5.1. Question Preparation

An examiner (HMA collector) prepares a set of questions for an examination. For our purpose, it is useful that the questions are designed so that various HMAs can be collected. This set of questions is stored under the Extensible Markup Language (XML) file format, as shown in Figure 3.1. Each paragraph in the tag name “question\_t” corresponds to the tenth question. Our tool uses an open-source display engine MathJax, which receives a question including LaTeX as an input and outputs an HTML, a Scalable Vector Graphics or a MathML. Then, a web browser panel loads and displays the question in the question window of our interface. Due to this setting, the examiner can use any character, symbol, or format that is supported by LaTeX in the end-user device. Moreover, he/she might set a time limit for participants to take the exam.

### 3.5.2. Collection Process

We assume the following steps to collect HMAs from each participant.

**Step 1:** We explain the purpose of our research and ask a participant whether he/she agrees to contribute. We also explain how the e-testing tool collects, stores, uses, and shares his/her HMAs.

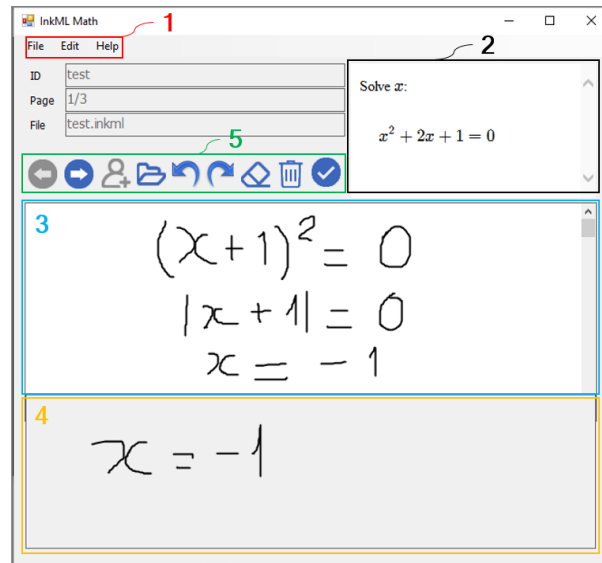


Figure 3.2. Main interface of our e-testing tool.

**Step 2:** If agreed by the participant, the tool collects personal information about the participant including gender, age, dominant hand, writing hand, occupation, and mother tongue in the same way as collections of previous online handwriting databases [53–55]. This information is only used for academic purposes, and some of them will be eliminated when publicly releasing the dataset.

**Step 3:** The participant selects a set of questions prepared by the examiner.










**Step 4:** Each question is displayed in Window 2 in Figure 3.2. The participant can write intermediate progress in Window 3 and the final answer in Window 4 in Figure 3.2. When the participant completes writing their answer for the current question or skips the question to do later, he/she can easily move to another question. This step is repeated until finishing all the questions.

### 3.5.3. Design Considerations

The aim of using two writing windows is to develop two kinds of datasets which allow researching two levels of answers for questions, full answers, and short answers. This collecting strategy is designed not only for math but also for analysis-required subjects.

Our collection tool is designed and implemented for recording and storing the pen trajectory in the entire writing area via ink-space (HIMETRIC) by the unit of 0.01 mm, which is independent of the display resolution. Thus, the tool can be used on devices with different resolutions. This collection tool records not only the pen trajectory but also the stroke duration time and pressure information. To support participants, we provide nine necessary functions which are presented in Table 3.1.

Table 3.1. A Description of available functions in rectangle 5 of Figure 3.2.

#	Icon	Description
1		Display the previous question with the handwritten answer (if existing).
2		Display the next question with the handwritten answer (if existing).
3		Create a new participant's profile for collecting new patterns.
4		Open an existing answer file
5		Erases the last change done in the current writing windows to an older state
6		Reverses the undo.
7		A deletion mode that works as an eraser for removing strokes the computer pointer touches while holding the left mouse.
8		Erase all strokes in the current writing windows.
9		Finish the current collecting section and save HMAs.

### 3.6. HMA Annotation Process

This section describes the process to mark or annotate HMAs and the output format of collected HMAs.

#### 3.6.1. Annotation Process

Figure 3.3 and Figure 3.4 show the displays of the e-marking tool. A user as a marker takes the following steps to mark participants' answers. Our tool is designed to allow the marker to mark answers for each question.

**Step 1:** Load HMA files containing participants' answers. Then, a list of questions in the selected files is displayed on Window 1 in Figure 3.3. To view the state of the marking progress, the number of marked answers to each question is shown under text-based and color-based displays.

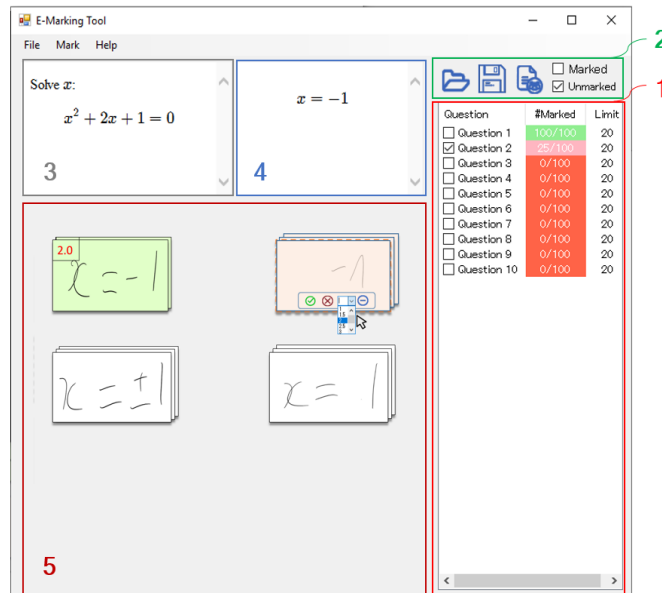


Figure 3.3. Main interface for marking multiple HMAs.

**Step 2:** Select a question to mark by checking the corresponding box of the question. Then, Windows 3 and 4 present the question and its specimen answers, respectively.

**Step 3:** Mark answers. In our e-marking tool, a clustering algorithm is applied for grouping similar short HMAs together, and then the marker could take a single action to mark a group of answers. Since the number of answers might be too large for clustering and marking at once time, the marker should set this number by directly inputting the limit into the third column corresponding to the question. Besides, a default limitation of the quantity can be set, as shown in Figure 3.3. After executing the clustering function by utilizing the button #3 in Table 2, Window 5 displays multiple short answers to the selected question in many groups. The marker taps on each group of answers to view the clustering result. If they are well clustered, he/she can mark each group at the same time. Otherwise, some HMAs outside are included in the group, an incremental refinement approach proposed in [5] could be applied to re-assigned them into other groups. In case that the short answer of an HMA is not clear, the marker can tap on the answer to view

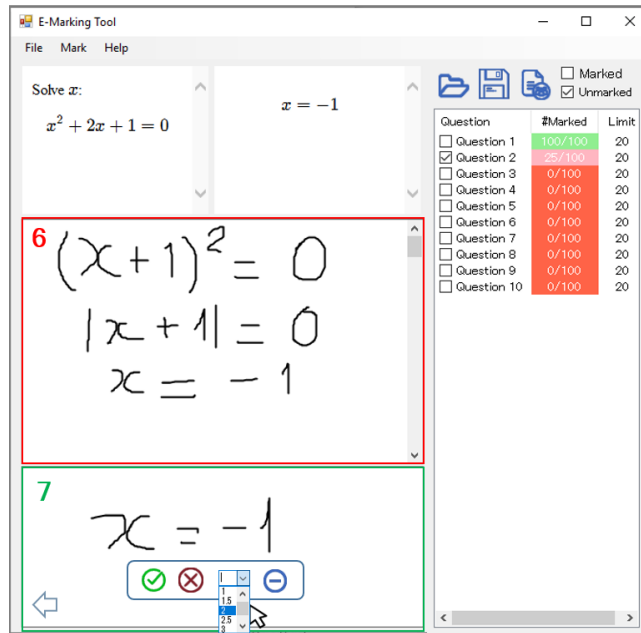


Figure 3.4. Main interface for marking a single HMA.

the full answer carefully and mark it by comparing the participant’s answer and the specimen. An annotation as “correct” or “incorrect” can be tagged to each HMA by using buttons #4 and #5 described in Table 3.2. He/she also can input an intermediate score by using a drop-down list. To remove the tag or the score, the marker can use button #6. Besides, two checkboxes in Window 2: “marked” and “unmarked”, work as filters of marked and unmarked questions to review and mark HMAs, respectively. The whole step is repeated until the last selected answers.

### 3.6.2. Output Structure and Format

Table 3.2. A description of available functions in Figure 3.3 and Figure 3.4

#	Icon	Description
1		Load participant’s answers
2		Save participants’ file containing marked answers to a specified folder
3		Execute clustering to group similar answers
4		Mark the current answer with “correct” tag
5		Mark the current answer with “incorrect” tag
6		Clear the annotation tag or the intermediate score of the current answer

This section presents the basic structure of the collected data. For each examination, a specific folder is created. The disclosable profile of the examiner is recorded as well as the set of questions. Then, for each examinee, a specific folder inside the examination folder is created. In each sub-folder, the disclosable profile of the examinee and the tablet used are recorded. Then, all his answers to questions are recorded in InkML. Offline patterns can be rendered from the corresponding online patterns.

## **CHAPTER 4. Clustering online handwritten mathematical answers**

### **4.1. Introduction**

We present two approaches for clustering OHMEs. The first approach is to extract features from each OHME and apply a clustering algorithm on the set of extracted features. We extract features from low-level pattern features to high-level symbolic and structural features obtained from processing and recognizing OHMEs. We employ bag-of-features composed of low-level directional features and high-level recognition-based features, i.e., bag-of-symbols, bag-of-relations, bag-of-positions, position-based bag-of-symbols, and position-based bag-of-relations. Low-level features are free from recognition accuracy but are not robust to various ways of writing an ME. Features from several levels of OHME recognition may be fragile due to its immaturity, but they may provide useful distinctive features. To reduce the dimensionality of our proposed feature spaces, we present distance-based representation (DbR). We also consider a method for combining these types of features to improve the performance.

The second approach is to compute pairwise similarities among OHMEs. We present a method that utilizes a generative sequence similarity function (GSSF) and a data-driven representation for each OHME. GSSF is formed by high-level sequential features, probability terms of the output sequence generated from a sequence-to-sequence (Seq2Seq) OHME recognizer. The sequential features are dynamic, and they could represent the global structure of OHME. Each OHME is then represented by a vector of similarity scores with other OHMEs, namely similarity-based representation (SbR). SbR allows controlling the dimensionality of the feature space to reduce the influence of the concentration phenomenon. Finally, we input the SbR matrix into a clustering algorithm such as k-means to obtain the clusters of OHMEs.

The rest of this chapter is organized as follows. Section 4.2 introduces our proposed methods in detail. Section 4.3 presents problems related to the cost of a clustering-based marking. Section 4.4 presents our experiments for evaluating the proposed methods. Finally, section 6 concludes the work.

### **4.2. Our proposed methods**

We first present the approach of extracting multi-level features from HMEs. Then, we present the second approach which utilize a pairwise similarity function. Finally, we present measurements for clustering-based marking proposed in [13].

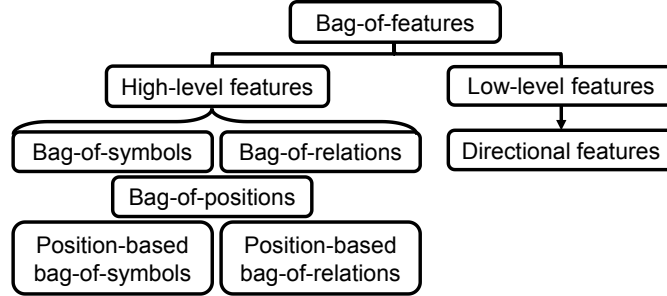


Figure 4.1. Types of features.

#### 4.2.1. Multi-level features from OHMEs

We propose two types of features, i.e., low-level pattern features and high-level symbolic and structural features, as shown in Figure 4.1.

##### 4.2.1.1. Low-level features

Low-level features are extracted from patterns without their recognition and interpretation. To avoid stroke direction and order variations, we convert OHMEs to offline images (denoted as OHME images) and extract image-based directional features of all strokes. These are common features for offline handwritten character recognition [58] [59].

The feature extraction process consists of three steps, as shown in Figure 4.2. Three main steps for extracting directional features, i.e., non-linear normalization, directional decomposition, and Gaussian filtering and feature assembly. To recognize a single character image, it is usually normalized into a fixed-size box and four or eight directional features are extracted and partitioned into a fixed-size grid of regions. Since we want to handle various sizes of OHMEs instead of characters or symbols, the size of the normalized images and the grid sizes are adapted with respect to each OHME. On the other hand, feature vectors with a fixed length are convenient for comparison. Therefore, we use the average height and width of all input OHME images denoted as  $\bar{H}$  and  $\bar{W}$ , respectively, to normalize them into the same dimension. For each OHME image, we perform directional decomposition and divide each decomposed direction into  $R \times C$  partitions such that each region captures at most one symbol. This is expected to represent directional features effectively for most input OHME images although it is difficult for some. We set  $R$  and  $C$  as follows:

$$R = \frac{\bar{H}}{\bar{H}_{max}}, C = \frac{\bar{W}}{\bar{W}_{max}} \quad (1)$$



where

$$\bar{H}_{max} = \max(\bar{H}_{CC}^{I_1}, \bar{H}_{CC}^{I_2}, \dots, \bar{H}_{CC}^{I_N})$$

$$\bar{W}_{max} = \max(\bar{W}_{CC}^{I_1}, \bar{W}_{CC}^{I_2}, \dots, \bar{W}_{CC}^{I_N})$$

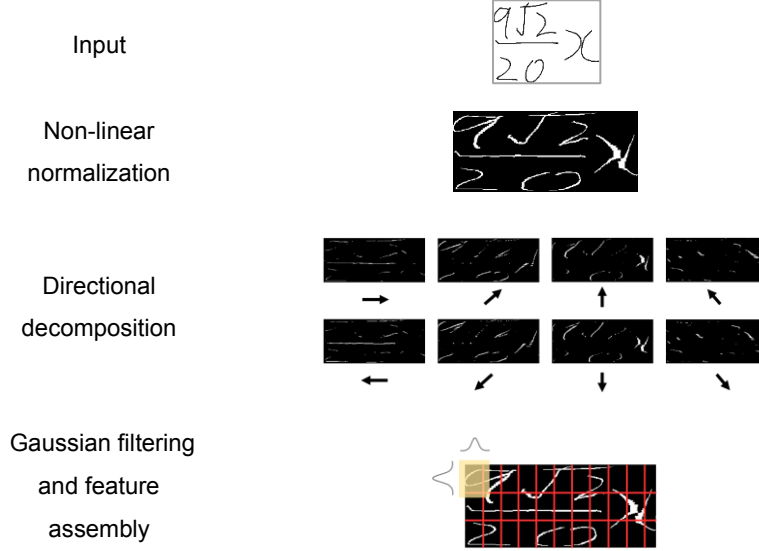


Figure 4.2. Three main steps for extracting directional features.

$I = \{I_1, I_2, \dots, I_N\}$  is the set of normalized HME images and  $\bar{H}_{cc}^{I_i}$  and  $\bar{W}_{cc}^{I_i}$  (CC: connected component) are the average height and width of the connected components in each  $I_i$ , respectively.

Then, we blur the partitioned regions using a low-pass Gaussian filter with size  $(H_{grid} + W_{grid}) \times (H_{grid} + W_{grid})$ , where  $H_{grid}$  and  $W_{grid}$  are the height and width of the partitioned regions, respectively. Finally, we obtain a feature vector with length  $R \times C \times 8$  by taking the Cartesian product of all columns and all rows.

#### 4.2.1.2. High-level features

High-level features are symbolic and structural features obtained by OHME recognition and represented by a symbol relation tree (SRT). An SRT is a directed graph representing symbols (nodes) and spatial relationships (edges) between two symbols in an OHME, as shown in Figure 4.3. Since an SRT carries rich information, it is difficult to represent it entirely on a vector space. One way to achieve this is to present each kind of information in the SRT separately and then combine them. In this study, we decompose the information into 4 types of features:

- Bag-of-symbols (BoS): occurrences of symbols.
- Bag-of-relations (BoR): occurrences of spatial relationships.
- Bag-of-positions (BoP): occurrences of partitioned positions having symbols.

- Position-based BoS (PbBoS) and position-based BoR (PbBoR): BoS and BoR extracted within each partitioned position and aggregated from all of them, respectively.

We can capture these features from several candidates of recognition, but we select the top candidate to extract them.

### A. Bag-of-symbols

The bag-of-words model and its enhancements have been demonstrated to be efficient for representing documents in text classification [60], natural language processing [61], and document clustering [62]. For OHME clustering, the occurrences of symbols play an important role in clustering. In this work, we use BoS to represent how often each symbol appears in an OHME. Then, we put it into a vector representing the frequencies of appearances in an available list of symbols, as shown in Figure 4.3.

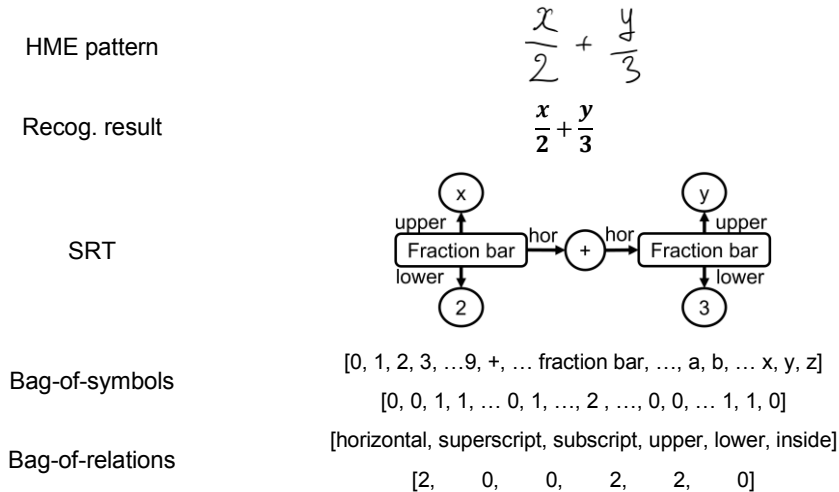
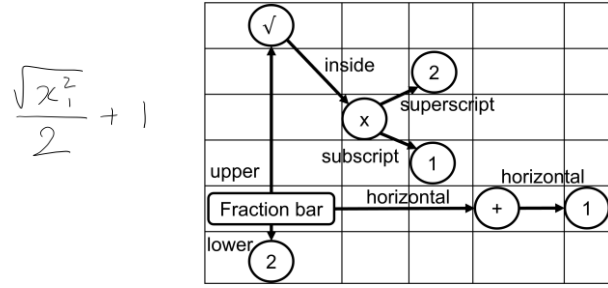


Figure 4.3. Bag-of-symbols and bag-of-relations for a given OHME, where “recog.” and “hor” are abbreviations of “recognition” and “horizontal” respectively.

### B. Bag-of-relations

We consider six types of spatial relationships between symbols, i.e., horizontal, superscript, subscript, upper, lower, and inside, as shown in Figure 4.4. BoR represents how many of each type of spatial relationship occurs in an OHME in the form of a vector.

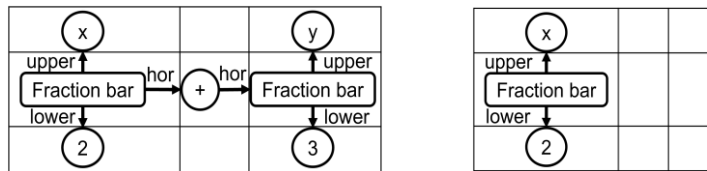


(a) HME (b) SRT and its division into  $M \times N$  positions.

Figure 4.4. Example for dividing SRT consisting of the six types of spatial relationships.

### C. Bag-of-positions

While BoS and BoR reflect information about symbols and spatial relationships, BoP represents the occurrences of some symbols in partitioned positions in an SRT. An SRT is divided into  $M \times N$  positions, each of which is expected to contain at most one symbol. This division also reflects the spatial relationships among symbols. An example for dividing SRT is shown in Figure 4.4(b). The parameters,  $M$  and  $N$ , could be set as the size of SRT. However, because the size of SRT depends on each ME, we cannot generate the feature vectors in the same dimension. To address this issue, we use the size of the largest SRT among OHMEs to normalize others, as shown in Figure 4.5. Then, the partitioned position containing a symbol is marked 1 and 0, otherwise, and expressed by a matrix  $P_{M \times N} = (P_{ij})$  with  $P_{ij} \in \{0, 1\}$ . To make these features more robust, we apply a Gaussian filter with size of  $3 \times 3$  for blurring. Finally, a feature vector is formed by taking the Cartesian product of all columns and all rows.



(a) Largest SRT

(b) Case of a simple SRT

Figure 4.5. Illustration of using the largest SRT to divide others.

### D. Position-based bag-of-symbols and bag-of-relations

In BoS and BoR, the captured information is discrete and does not explicitly express the structure of an OHME. Besides, BoP only provides features of symbols' positions without mentioning the classes of symbols. Therefore, we add another type of features while extracting symbols and spatial relationships, as shown in Figure 4.6. Firstly, we divide the SRT into  $M \times N$  positions as mentioned in the previous section. Then, from each partitioned position, we extract BoS and BoR. We extend the position to its neighbors or perform zero-padding to obtain  $3 \times 3$  positions. We convolute the values of  $3 \times 3$  positions with the  $3 \times 3$  Gaussian mask and take the sum of all the values for the position.

We finally obtain two types of feature vectors, i.e., PbBoS and PbBoR, by taking the Cartesian product of all columns and all rows for each type of features.

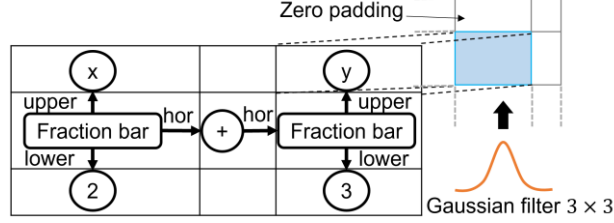


Figure 4.6. Illustration of dividing an SRT into  $M \times N$  positions, performing zero padding, and applying a Gaussian filter over the position and its neighbors.

#### 4.2.1.3. Feature combination

A single type of bag-of-features may not contain enough information for clustering; hence, we combine multiple types to improve the performance. That is, we concatenate multiple feature vectors into a single vector. However, this method may become ineffective because different feature vectors have different meanings and belong to separate spaces. Here, we use weighting parameters to optimize the method, where the distance between two samples is calculated according to Eq. (2):

$$Dist(OHME_1, OHME_2) = \sum_{i=1}^I \alpha_i d(f_i(OHME_1), f_i(OHME_2)) \quad (2)$$

where  $OHME_1$  and  $OHME_2$  are two OHMEs to compute the distance,  $f_i$  is a type of features,  $I$  is the number of types of features,  $d(v_1, v_2)$  is the Euclidean distance between two vectors  $v_1$  and  $v_2$ , and  $\alpha_i > 0$  is a weighting parameter for each type of features satisfying  $\sum_{i=1}^I \alpha_i = 1$ . These parameters are determined by applying the enumeration method with  $\alpha_i \in \{0.1, 0.2, 0.3, \dots, 0.9\}$ .

#### 4.2.2. Distance-based representation

High-level recognition-based features composed of the occurrences of symbols and spatial relationships appearing in an OHME make their feature space sparse, especially for PbBoS and PbBoR. For example, with a dictionary of 101 math symbols, the length of PbBoS of the expression “a+b” is 303 ( $M \times N \times 101 = 1 \times 3 \times 101$ ), but only 5 values in this feature vector are non-zero after applying the Gaussian filter. In practice,  $M$  and  $N$  could be large when MEs are complex. According to the study in [63], all  $p$ -norms  $\|X\|_p = (\sum_i |X_i|^p)^{1/p}$  ( $p > 0$ ) including the Euclidean distance ( $p = 2$ ) seem ineffective for high-dimensional data due to the concentration phenomenon, where all distances among pairs of data points seem to be very similar. Hence, the sparse features and high dimensionality of combined features might result in low performance when using the  $p$ -norms. Inspired by the idea of using dissimilarities of each OHME to all other OHMEs as a representation for the OHME [14], we similarly calculate the distance from

each OHME's feature vector to all other OHMEs' feature vectors to form distance-based representation (DbR). OHMEs belonging to the same cluster could produce similar DbR while those belonging to different clusters could not so that DbR could be used for clustering. Moreover, we expect that DbR containing pairwise distances among the feature vectors is informative, it has a lower dimensional space, and it is less sparse than the original feature space. Given a set of  $N$  OHMEs  $\{X_1, X_2, \dots, X_N\}$ , DbR of  $X_i$  is calculated according to Eq. (3):

$$DbR(X_i) = \{D_{i,j}\}_{j=1,N} \quad (3)$$

where

$$D_{i,j} = d(f(X_i), f(X_j)) \quad (4)$$

$d(v_1, v_2)$  is the distance between two vectors  $v_1$  and  $v_2$ ,  $f$  is a feature mapping function. Note that the dimensionality of DbR depends on the number of OHMEs. If this number is large, we also obtain a high-dimensional feature space. However, markers can divide a large group of OHMEs into small ones, then they can apply clustering on each of them to avoid this problem in practical cases. Also, we may apply mini-batch clustering. In our experiments, we show that DbR improves the clustering performance in almost all cases even though we set the distance function  $d$  as the Euclidean distance to compute the pairwise distances.

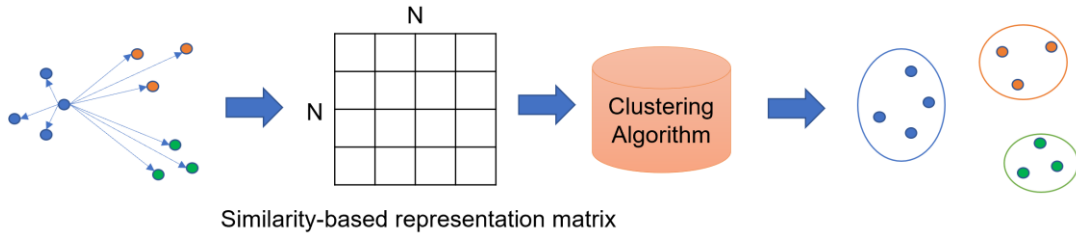


Figure 4.7. Clustering process with pairwise similarity function.

### 4.2.3. Generative sequence similarity function based on a Seq2Seq model

Cluster analysis is useful for exploratory analysis by partitioning unlabeled samples into meaningful groups. For this problem, we traditionally extract useful features from each sample and pass them to a clustering algorithm such as k-means to partition them into groups. Here, we utilize a type of data-driven representation for each sample. We represent each sample by pairwise similarities between it and other samples. This idea is the same as the distance-based representation presented in section 4.2.2. Then, we form a similarity-based representation (SbR) matrix. The SbR matrix is inputted to a clustering algorithm to obtain clusters of OHMEs. The overall process of our proposed method is shown in Figure 4.7. This section presents our proposed similarity function (SF), then describes the SbR.

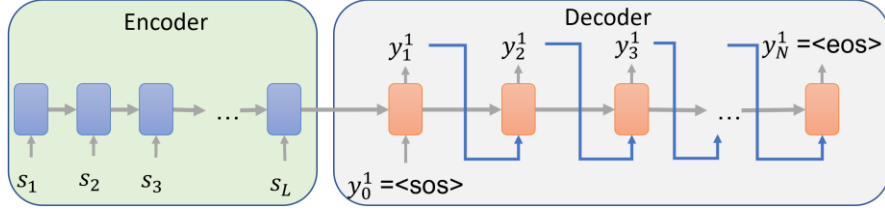


Figure 4.8. A standard Seq2Seq model.

#### 4.2.3.1. Generative sequence similarity function

Our proposed SF gives a similarity score between two OHMEs based on a Seq2Seq OHME recognizer. We expect that the similarity score of two OHMEs containing the same ME is significantly higher than those with different MEs.

A standard Seq2Seq model, as shown in Figure 4.8, consists of two parts: an encoder that receives an input sequence to represent high-level features and a decoder that sequentially generates an output sequence from the encoded features and the previous prediction. Given two input OHMEs denoted as  $S_1$  and  $S_2$ , the recognizer generates LaTeX sequences of  $\{y_i^1\}_{i=1,N}$  and  $\{y_j^2\}_{j=1,M}$ , where  $y_i^1$  and  $y_j^2$  are symbol classes in the vocabulary, and  $N$  and  $M$  are the lengths of the output sequences.

A simple idea to form the similarity score of two OHMEs is to calculate the edit distance of the two output sequences  $\{y_i^1\}_i$  and  $\{y_j^2\}_j$ . However, this method might not be effective since the edit distance only utilizes the differences in terms of recognized symbol classes, but the probabilities of recognized symbols seem more important than the symbol classes. Our proposed SF utilizes terms of probabilities of recognized symbol classes instead of the symbol classes.

Another difficulty of directly comparing two output sequences or generated probabilities is that their lengths are variant. The proposed SF uses the symbol predictions of an OHME to input into the decoder of another OHME for computing the terms of probabilities. Those probabilities are formed on the output sequence of one of those two OHMEs. Hence, the SF is not influenced by the size-variant problem.

Our SF consists of two main components: the similarity score of  $S_1$  compared to  $S_2$  and the one of  $S_2$  compared to  $S_1$  denoted as  $F(S_1|S_2)$  and  $F(S_2|S_1)$ , respectively. We firstly define  $F(S_1|S_2)$  as follows:

$$F(S_1|S_2) = \sum_{i=1}^N \left( \log \left( P(y_i^1|S_2, y_{i-1}^1) \right) - \log \left( P(y_i^1|S_1, y_{i-1}^1) \right) \right) \quad (5)$$

where  $P(x|y, z)$  is the probability of  $x$  given  $y$  and  $z$ . An illustration of computing  $F(S_1|S_2)$  is shown in Figure 4.9. The predicted symbol  $y_i^1$  at the  $i$ -th time step is inputted

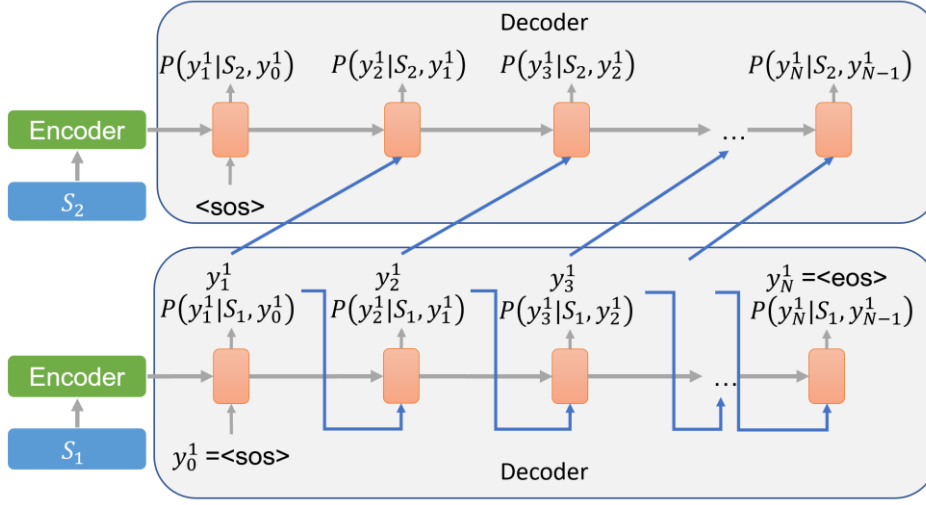


Figure 4.9. Illustration of computing  $F(S_1|S_2)$ .

to the  $(i+1)$ -th time step of the  $S_2$  decoder for computing the probability  $P(y_i^1|S_2, y_{i-1}^1)$ . Similarly, we define  $F(S_2|S_1)$  as follows:

$$F(S_2|S_1) = \sum_{j=1}^M \left( \log \left( P(y_j^2|S_1, y_{j-1}^2) \right) - \log \left( P(y_j^2|S_2, y_{j-1}^2) \right) \right) \quad (6)$$

The  $F$  function is not appropriate for the clustering algorithms such as  $k$ -means because it is asymmetrical. Thus, we compute an average of  $F(S_1|S_2)$  and  $F(S_2|S_1)$  that is symmetrical measurement. We name it as the Generative Sequence Similarity Function (GSSF), which is computed as follows:

$$\text{GSSF}(S_1, S_2) = \frac{F(S_1|S_2) + F(S_2|S_1)}{2} \quad (7)$$

Assume that the Seq2Seq recognizer are well recognized  $S_1$  and  $S_2$ . GSSF has some properties as follows:

- $\text{GSSF}(S_1, S_1)$  equals to zero if and only if  $F(S_1|S_1)$  equals to zero.
- $\text{GSSF}(S_1, S_2)$  is approximately zero if  $S_1$  and  $S_2$  denote the same ME. In this case,  $F(S_1|S_2)$  and  $F(S_2|S_1)$  are both around zero.
- $\text{GSSF}(S_1, S_2)$  is negative if  $S_1$  and  $S_2$  denote two different MEs. In this case, both  $F(S_1|S_2)$  and  $F(S_2|S_1)$  are much lower than zero.
- GSSF is a symmetric function.

#### 4.2.3.2. Similarity-based representation

Given  $N$  OHMEs  $\{X_1, X_2, \dots, X_N\}$ , SbR of  $X_i$  is formed by a pre-defined pairwise SF:

$$\text{SbR}(X_i) = [\text{SF}(X_i, X_1), \dots, \text{SF}(X_i, X_i), \dots, \text{SF}(X_i, X_N)] \quad (8)$$

### 4.3. Measurements for clustering-based marking

In clustering-based marking systems, a human marker marks the major set of answers for each cluster collectively and selects the minor ones for manual marking separately. Hence, the cost of the marking process depends on how many samples belong to the major set and how few answers in the minor sets are included in each cluster. For this reason, we measure purity to evaluate the performance of the clustering task as shown in Eq. (9):

$$Purity(G, C) = \frac{1}{H} \sum_{k=1}^K \max_{1 \leq i \leq J} |g_k \cap c_i| \quad (9)$$

where  $H$  is the number of samples,  $J$  is the number of categories (the right number of clusters),  $C = \{c_1, c_2, \dots, c_J\}$  is a set of categories,  $K$  is the number of clusters, and  $G = \{g_1, g_2, \dots, g_K\}$  is a set of obtained clusters.

However, high purity is easy to achieve when the number of clusters is large. For example, when  $K$  is equal to  $H$ , we obtain a perfect purity of 1.0. Hence, we set the number of clusters as the number of categories to evaluate in our experiments.

The purity alone does not show the quality of clustering in the clustering-based marking systems. We employ a cost function presented in Khuong et al. [13], reflecting a scenario of verifying and marking answers in the clustering-based marking systems. For each cluster, the verifying task is to find a major set of answers by filtering minor answers, while the marking task is to compare the major set and minor answers with the correct answer or the partially correct answers. The marking cost ( $MC$ ) is composed of the verifying time  $C_{ver}$  and the marking time  $C_{mark}$  as shown in Eq. (10):

$$\begin{aligned} f(G, C) &= \sum_{i=1}^K cost(g_i, C) = \sum_{i=1}^K (C_{ver}(g_i, C) + C_{mark}(g_i, C)) \\ &= \sum_{i=1}^K (|g_i| \times \alpha T + (1 + |g_i| - |M_i|) \times T) \end{aligned} \quad (10)$$

where  $T$  is the time unit to mark an answer. There exists a real number  $\alpha$  ( $0 < \alpha \leq 1$ ) so that the verifying cost of an answer is  $\alpha T$ .  $C_{ver}(g_i, C) = |g_i| \times \alpha T$  is the cost of verifying all answers in the cluster  $g_i$ .  $C_{mark}(g_i, C) = (1 + |g_i| - |M_i|) \times T$  is the cost of marking the major set of answers  $M_i$  and all minor answers in the cluster  $g_i$ . For simplicity, we assume  $\alpha = 1$ , implying that the verification time is the same as the marking time  $T$ . We normalize Eq. (10) into  $[0, 1]$ , and obtain Eq. (11) as follows:

$$MC(G, C) = \frac{f(G, C)}{2NT} = \frac{K}{2H} + \left(1 - \frac{1}{2} Purity(G, C)\right) \quad (11)$$

$MC$  equals 1 in the worst case if the number of clusters equals the number of answers. It implies that  $MC$  approaches the cost of marking all answers one-by-one.



Table 4.1. Details of the Dset\_50 dataset.

Dset_50	
# Students	21
# OHMEs categories	50
# Interface styles	3
Total # of OHMEs	$21 \times 50 \times 3 = 3150$

## 4.4. Experiments

In this section, we present evaluations for our two proposed approaches.

### 4.4.1. Experiments on multi-level features of OHMEs

This section presents an evaluation on multi-level features of OHMEs on two datasets: Dset\_50 and Dset\_Mix.

#### 4.4.1.1. Datasets

We use two datasets of OHMEs to evaluate our proposed method. The first dataset, named Dset\_50, was collected from 21 students. Each student wrote 50 OHMEs three times on three kinds of writing interfaces (i.e., without any guiding line, with a centerline, and with the center, top, and bottom lines). As a result, the total number of online OHMEs is 3150 samples, which belong to 50 classes. This dataset contains common symbols that belong to 101 classes used in the CROHME 2019 competition. Table 4.1 provides the details of the Dset\_50 dataset.

The second dataset, named Dset\_Mix<sup>1</sup>, has mixed patterns of real (genuine) OHMEs from CROHME 2016 and synthesized patterns made from LaTeX sequences and isolated handwritten symbol patterns from CROHME 2016. Note that collecting real answers from students in a real examination is ideal but it requires agreement among all participants, teachers, and schools. Hence, we generated synthesized answers and publish this dataset for the research community to use [64].

---

<sup>1</sup>Uploaded at: [http://tc11.cvc.uab.es/datasets/Dset\\_Mix\\_1](http://tc11.cvc.uab.es/datasets/Dset_Mix_1)

Table 4.2. Details of the Dset\_Mix dataset.

# Categories and patterns	Subgroup No.									
	1	2	3	4	5	6	7	8	9	10
# Categories of correct answers	2	1	1	1	1	1	2	1	1	1
# Genuine patterns of correct answers	26	0	0	20	20	20	20	27	0	0
# Synthesized patterns of correct answers	21	40	50	20	35	10	81	49	50	50
# Categories of incorrect answers	8	4	5	4	4	6	2	5	3	3
# Genuine patterns of incorrect answers	21	18	3	19	2	39	1	27	0	0
# Synthesized patterns of incorrect answers	132	142	147	141	143	131	98	97	150	150
# Total answers	200	200	200	200	200	200	200	200	200	200

Dset\_Mix stores ten subgroups corresponding to ten questions. Each subgroup consists of 200 OHMEs, which is a mixture of genuine patterns and synthesized patterns and a few correct answers and several incorrect answers for a math question. Note that OHMEs in each subgroup are very similar to those in Dset\_50. The sample size for each question is set based on the number of students in each grade of common schools. Table 4.2 provides the details of the Dset\_Mix dataset.

The synthesized OHMEs were generated according to the method proposed in [65]. This method consists of three main steps. Given a LaTeX sequence or a MathML script, the method firstly generated a template that presents the sizes and positions of the symbols based on their spatial relationships to each other (i.e., horizontal, superscript, subscript, upper, lower, and inside). Secondly, the method made the generated patterns look more natural by randomly changing the sizes and positions of the symbols slightly. Finally, the isolated handwritten symbol patterns in CROHME 2016 are filled in the generated template. Figure 4.10 shows some samples in subgroup 6 of Dset\_Mix.

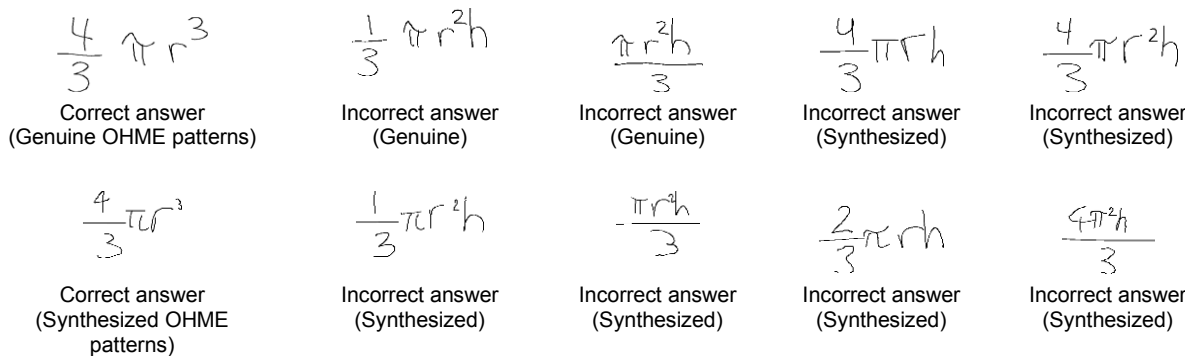


Figure 4.10. Samples in subgroup 6 of Dset\_Mix.

#### 4.4.1.2. Online HME recognizer

In our previous work [57], we used the OHME recognition engine by Le et al. [31]. In this thesis, we employed its enhanced version with a significantly better recognition rate.

In our enhanced engine, we improved the symbol classification with bidirectional context by utilizing a deep bidirectional long short-term memory (BLSTM) and connectionist temporal classification (CTC) [66]. The BLSTM model consists of three BLSTM layers with 128 hidden units for each layer. Four basic point-based features, i.e., the sine and cosine of writing directions, the normalized distance between the preceding and the succeeding of the current point and the binary value of pen state were extracted from each point of an OHME. Then, a sequence of point-based features was input to the BLSTM model and the CTC decoder to produce a sequence of classification probabilities corresponding to the input sequence. Next, the hypotheses of symbol segmentation and symbol recognition were formed from the obtained probabilities. Finally, Stochastic Context-Free Grammar (SCFG) with a list of predefined grammar rules was applied for producing the recognition result based on these hypotheses obtained. By the context information, it is expected to solve several ambiguous cases that the previous version could not handle. We call the method Context SCFG (C\_SCFG). Moreover, an n-gram language model was applied as part of the post-processing of the top-5 best recognition candidates and a large number of grammar rules was added to help the engine cover more math expressions. The engine and the n-gram model were separately trained on the CROHME 2016 training set and the corpus of LaTeX formulas from English Wikipedia provided in CROHME 2016 [67], respectively. The detail of this recognizer is presented in section 5.2.1.

Table 4.3 shows an improvement of the C\_SCFG engine in terms of the average value of the expression recognition rates, the F1-score of symbol recognition, and the F1-score of spatial relationship recognition on the CROHME 2016 testing set, Dset\_50, and Dset\_Mix. Because the parameters for combining features need to be trained, Dset\_50 was divided into 5 subsets, with each engine trained by 4 subsets and tested for the remaining subset; then, the performance for the 5 subsets are averaged (5-fold cross-validation). Thus, values in each row for Dset\_50 correspond to averages with standard deviations for each engine. On the other hand, values for Dset\_Mix correspond to averages and standard deviations among its subgroups.

We obtained these values by using the LgEval tool proposed in [68]. Compared with the state-of-the-art end-to-end OHME recognition system [40], which is an extension of the record-breaking end-to-end OHME recognition method named TAP [27], our C\_SCFG engine is 3.75% point better in expression rate on the CROHME 2016 testing set.

#### **4.4.1.3. Experiment settings**

Experiments was conducted on an Intel Xeon CPU@3.30GHz Desktop PC.

Table 4.3. Expression recognition rate and F1-score of symbol and spatial relationship recognition.

Version	Dataset	Expression rate (%)	F1-score (%)	
			Symbols	Spatial Relationships
Le et al. [58]	CROHME 2016	43.94	65.81	83.27
	Dset_50	71.65±5.67	85.94±2.90	95.07±1.52
	Dset_Mix	14.70±21.86	78.79±11.88	75.58±21.54
C_SCFG	CROHME 2016	51.70	74.30	85.99
	Dset_50	75.62±3.34	90.38±2.15	96.72±0.99
	Dset_Mix	46.9±24.73	78.56±15.97	82.74±17.66

We utilized the k-means algorithm for the clustering task using the Euclidean distance. In addition, we initialized centroids by using k-means++ [69], which is a popular variant of the k-means algorithm that tries to spread out initial centroids. To evaluate the proposed features, we set the number of clusters as the number of categories in our experiments.

To evaluate the performance of single types of features and their combinations, we implemented the experiments shown in Table 4.4 on the Dset\_50 and Dset\_Mix datasets. Then, we compared the results using the original features with those using DbR to cluster OHMEs. For DbR, we set the distance function  $d$  in Eq. (4) also as the Euclidean distance.

#### 4.4.1.4. Evaluation

We evaluated the performance of the clustering task when applying single types of features and combined features. Since this combination requires the weighting parameters to form the distance metric, we used a 5-fold cross-validation for Dset\_50 and Dset\_Mix in the same way as described in section 4.4.1.2. For feature combination, we firstly trained the combining parameters by using the original features, then applied the same parameters for DbR.

## A. Experiment on Dset\_50

Table 4.5 shows the purity for all the types of features and their combinations on the Dset\_50 dataset. The results show that we achieve the best value of purity around 0.992 when BoS with DbR is utilized. Moreover, BoS produces a better result compared with the other types of features. The feature type of spatial relationships alone (E3) or that of positions alone (E4) produces a low purity. We also find that adding positional information to BoS or BoR, i.e., PbBoS (E5) and PbBoR (E6), does not increase purity. Moreover, clustering with DbR yields better performance than using the original features.

Regarding feature combination, the combination of BoS, BoR, and BoP (E7) gives the highest purity when using DbR. Combining PbBoS and PbBoR (E8) does not increase purity. On the other hand, combining Dir with high-level features (E9, E10, and E11)

Table 4.4. Experiment settings on single types of features and their combinations

Feature type	Exp.	Dir	BoS	BoR	BoP	PbBoS	PbBoR
Single	E1	✓	-	-	-	-	-
	E2	-	✓	-	-	-	-
	E3	-	-	✓	-	-	-
	E4	-	-	-	✓	-	-
	E5	-	-	-	-	✓	-
	E6	-	-	-	-	-	✓
Feature Combination	E7	-	✓	✓	✓	-	-
	E8	-	-	-	-	✓	✓
	E9	✓	✓	✓	✓	-	-
	E10	✓	✓	✓	-	✓	✓
	E11	✓	✓	✓	✓	✓	✓

Table 4.5. Experiments on single types of features and their combinations for Dset\_50 and Dset\_Mix.

Feature type	Exp.	Dset_50			Dset_Mix	
		Original features		C_SCFG + DbR	Original features + C_SCFG	C_SCFG + DbR
		Old engine [58]	C_SCFG			
Single	E1		0.866±0.03	0.882±0.08	0.716±0.14	0.649±0.15
	E2	<b>0.871±0.02</b>	<b>0.919±0.02</b>	<b>0.992±0.01</b>	0.690±0.11	0.684±0.08
	E3	0.758±0.10	0.787±0.11	0.774±0.10	0.569±0.16	0.576±0.16
	E4	0.753±0.15	0.734±0.14	0.759±0.17	0.602±0.15	0.612±0.16
	E5	0.851±0.01	0.764±0.08	0.769±0.07	<b>0.729±0.15</b>	<b>0.696±0.15</b>
	E6	0.755±0.10	0.708±0.10	0.724±0.10	0.586±0.14	0.575±0.11
Feature Combination	E7	0.857±0.03	0.922±0.03	<b>0.990±0.01</b>	0.674±0.17	0.693±0.15
	E8	0.807±0.03	0.777±0.07	0.758±0.08	0.709±0.14	0.681±0.16
	E9	0.904±0.03	0.923±0.03	0.972±0.04	0.726±0.12	0.755±0.17
	E10	<b>0.919±0.02</b>	0.923±0.03	0.973±0.04	<b>0.764±0.11</b>	<b>0.777±0.17</b>
	E11	0.894±0.01	<b>0.933±0.03</b>	0.925±0.05	0.753±0.13	0.745±0.14

works well when using the original features, but it decreases purity when applying DbR. This phenomenon could be due to that BoS and DbR could achieve very high purity and Dir becomes redundant for this dataset. Moreover, DbR works better than the original features.

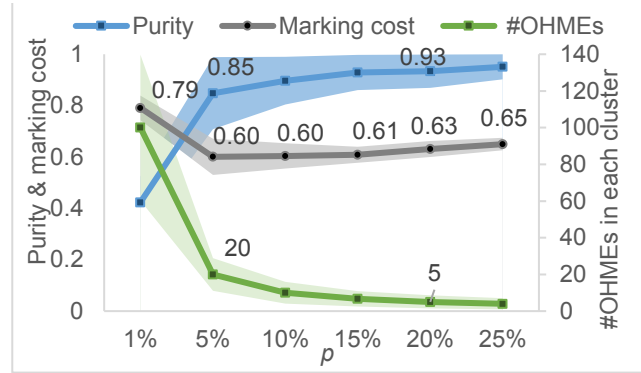
### **B. Experiment on Dset\_Mix**

Then, we evaluated our proposed method on the Dset\_Mix dataset. Table 4.5 shows the average values of purity on the 10 subgroups via a 5-fold cross-validation. Firstly, directional features (E1) yield a comparable result with using the original features. Secondly, within the single types of features, position-based BoS (PbBoS) in E5 achieves the best performance. Thirdly, feature combinations show better performance than individual types of features. Fourthly, DbR is not so efficient compared with the original features. Fifthly, purity is not improved without using the directional features as in E7 and E8. However, with directional features, purity in E10, E11, and E12 is increased significantly, where E10 yields the highest value of purity around 0.777 when using DbR. This demonstrates the importance of combining low-level features and high-level features. Overall, purity values obtained in this dataset are lower compared with those obtained in the Dset\_50 dataset.

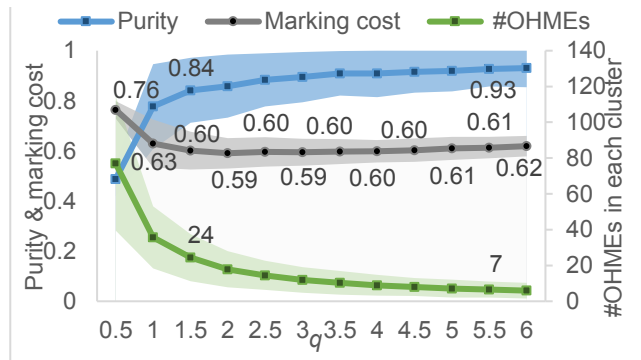
This is because OHMEs in each subgroup are more similar to one another than those in Dset\_50, which reflects the complication in realistic answers. Distances among OHMEs within the same category (intra-class distances) and distances among OHMEs belonging to different categories (inter-class distances) are not significantly different such that the DbR representation could not be effective.

Another reason for obtaining a low purity is that the number of clusters could be too small for a similar set of 200 answers for each question. If human markers viewed a smaller subset of answers without impure answers, they could mark each set more efficiently compared to marking a larger set of answers with several impure ones.

In practice, human markers do not know the exact number of clusters (denoted as  $k_{right}$ ) for a set of OHMEs. Hence, they may set the number of clusters (denoted as  $k$ ) as  $p$



(a) Setting  $k$  as  $p$  percent of the number of OHME patterns.



(b) Setting  $k$  as  $q$  times of  $k_{right}$ .

Figure 4.11. Average (lines) and standard deviation (light color areas) of purity, those of the marking cost, and those of #OHMEs in each cluster for Dset\_Mix with feature combination in

percent of the number of OHMEs (denoted as  $N$ ). Obviously,  $k$  should not be equal to  $N$  to benefit from clustering. To test this, we added another experiment on Dset\_Mix by using the setting that achieved the best performance on Dset\_Mix to present purity values when setting  $k$  as  $p$  percent of  $N$  and when increasing  $q$  times of  $k_{right}$ . To avoid  $k$  becoming large, we limited  $1\% \leq p \leq 25\%$  ( $2 \leq k \leq 50$ ) and  $1 \leq q \leq 6$  ( $4 \leq k \leq 60$ ). The average and standard deviation of purity, the marking cost ( $MC$ ), and the number of OHMEs of each cluster are shown in Figure 4.11. The results show that we could achieve purity in the range of  $[0.85 \pm 0.14, 0.93 \pm 0.06]$  and  $[0.84 \pm 0.13, 0.93 \pm 0.07]$  when  $p$  is in  $[5\%, 20\%]$  and  $q$  is in  $[1.5, 5.5]$ , respectively. Regarding  $MC$ , we could achieve  $MC$  in the range of  $[0.60 \pm 0.07, 0.63 \pm 0.03]$  and  $[0.60 \pm 0.08, 0.61 \pm 0.04]$  when  $p$  is in  $[5\%, 20\%]$  and  $q$  is in  $[1.5, 5.5]$ , respectively. With these settings of  $k$  according to  $p$  in  $[5\%, 20\%]$  and  $q$  in  $[1.5, 5.5]$ , the markers could benefit from clustering, where the clusters are quite pure, the average number of OHMEs in each cluster is in  $[20 \pm 9, 5 \pm 4]$  and  $[24 \pm 13, 7 \pm 4]$ , and  $MC$  is reduced by around  $[0.37, 0.4]$  and  $[0.39, 0.41]$  than manually marking OHMEs, respectively.

From this experiment, in the ideal case of the exact number of clusters, setting  $k$  as its  $q$  times in some range such as  $[2, 4.5]$  in Figure 4.11(b) would be slightly better than setting

Table 4.6. Results of  $k$  estimation and marking cost ( $MC$ ).

Subgroup	$k_{right}$	Silhouette		Hartigan	
		$k_{est}$	$MC$	$k_{est}$	$MC$
1	10	4	0.79	8	0.71
2	5	3	0.78	<b>5</b>	0.78
3	6	2	0.81	<b>6</b>	0.59
4	5	2	0.82	4	0.74
5	5	2	0.77	<b>5</b>	0.51
6	7	2	0.86	6	0.71
7	4	2	0.72	5	0.55
8	6	2	0.76	<b>6</b>	0.68
9	4	2	0.76	6	0.54
10	4	18	0.55	16	0.55
Avg. & Std. of $MC$		0.76±0.08		0.64±0.09	

(Avg.: average, Std.: standard deviation)

Table 4.7. Comparison with other research on Dset\_50 and Dset\_Mix.

Type	Method	Dset_50	Dset_Mix
Offline	Khuong et al. [13]	0.930±0.12	<b>0.830±0.07</b>
HMEs	Nguyen et al. [14]	0.98	*0.723±0.15
	DAC [74]	0.61	-
	Siamese Net [19]	0.79	-
OHMEs	<b>Our: E2 + C_SCFG + DbR</b>	<b>0.992±0.01</b>	0.684±0.08
	<b>Our: E10 + C_SCFG + DbR</b>	0.973±0.04	0.777±0.17

(\*: our retested result)

$k$  with  $p$  percent of the number of OHMEs. In practice, however,  $k_{right}$  is unavailable before marking, so that  $k$  could be chosen as some percent of  $N$ .

### C. Estimating the number of clusters

We also made an experiment to estimate  $k$  using two traditional indexes, i.e., Silhouette width [70] and Hartigan index [71]. Silhouette width is a ratio-type index measuring the ratio of within-cluster cohesion and between-cluster separation. Hartigan index is a heuristic rule of thumb based on the Euclidean within-cluster sum of squares. We estimated  $k$  values of subgroups on Dset\_Mix by using the setting that achieved the best performance on Dset\_Mix. For both methods, we searched for  $k$  in the range of [2, 20] for fair comparison. We measured the error of the estimation by using Relative Error ( $RE$ ) as shown in Eq. (12):

$$RE = \frac{|k_{est} - k_{right}|}{k_{right}} \quad (12)$$



where  $k_{est}$  is estimated by these two methods. The marking costs ( $MCs$ ) are presented in Table 4.6. According to  $k_{est}$ ,  $RE$  of Silhouette and Hartigan are  $0.38 \pm 0.14$  and  $0.19 \pm 0.34$ , respectively.  $RE$  and  $MC$  of the Hartigan index is significantly better than those of the Silhouette width. Although the variance of  $RE$  by Hartigan is large, its  $MC$  is acceptable. Moreover, Figure 4.11(a) implies that when  $k_{est}$  is in  $[10, 30]$ ,  $MC$  is still reduced by around  $[0.24, 0.41]$ . Therefore, we can apply the Hartigan index in practice. However, this  $k_{est}$  is not the best for our experiment. Heuristically assigning  $k$  according to  $p$  percent of  $N$  with  $p \in [5\%, 20\%]$  is better than  $k_{est}$  produced by the Hartigan index. Hence, we consider this problem as a remaining work.

#### 4.4.1.5. Comparing with other methods

We compare our results with recent research on clustering HMEs. So far, clustering OHMEs is not shown on the common dataset, Table 4.7 shows comparison with clustering offline HMEs converted from OHMEs in Dset\_50 and Dset\_Mix. Although our method is for online patterns, it performs better than the methods using bag-of-features [13], CNN-based features [14], Deep Adaptive Clustering (DAC) [72], and Siamese Net [19] on Dset\_50. However, the performance of our proposed method is still lower than the offline bag-of-feature [73].

#### 4.4.2. Experiments on generative sequence similarity function

This section presents the evaluation of our proposed SF on two answer datasets, i.e., Dset\_Mix and NIER\_CBT, by using the TAP recognizer proposed in [27] without the language model. We name this modified recognizer as MTAP.

##### 4.4.2.1. Datasets

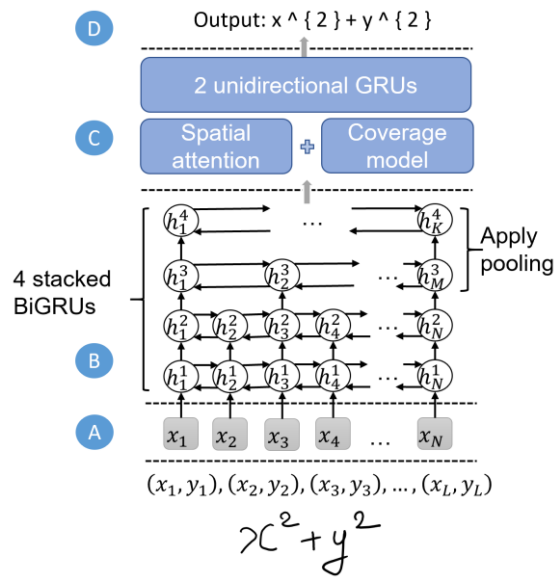


Figure 4.12. Overview of TAP consisting of point-based features as the input (A), the encoder part (B), the decoder part (C), and the output (D).

NIER\_CBT is a real answer dataset collected by a collaboration with National Institute for Educational Policy Research (NIER) in Tokyo, Japan. NIER carried out math tests for 256 participants, consisting of 249 students of grade 11 and 7 students of grade 7 at nine high schools. The participants answered a set of 5 questions within 50 minutes, then wrote their results on iPad by using an Apple pen and a developed tool with OHMEs captured. The details of the collection are presented in [74]. There are three sets of questions to obtain 934 answers for 15 questions. Since our OHME recognizer was trained for 101 common math symbols that appeared in the dataset of CROHME [1], we removed 15 OHMEs that contain out-of-vocab symbols. The number of correct/incorrect answer

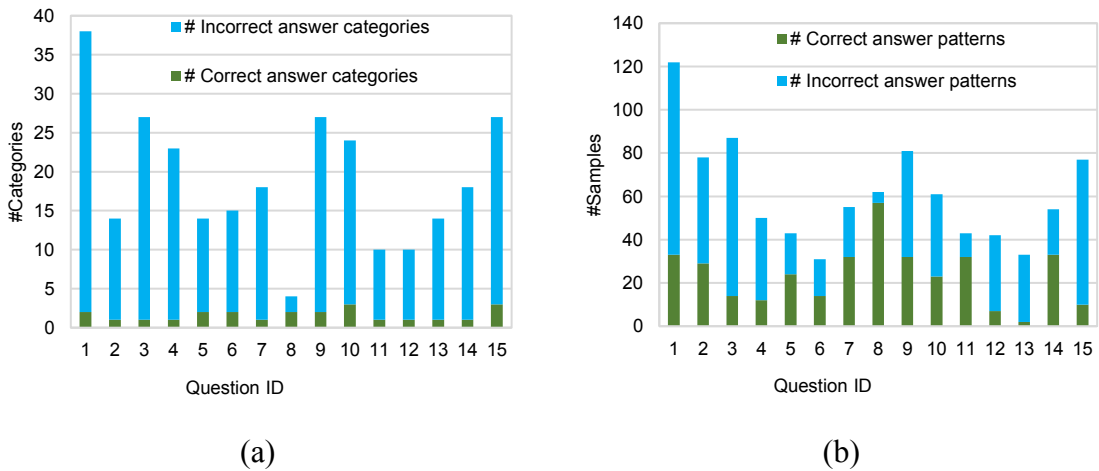


Figure 4.13. Details of NIER\_CBT. (a) shows the number of correct/incorrect categories in each question, and (b) shows the number of correct/incorrect patterns in each question.

categories and the number of correct/incorrect answer patterns in NIER\_CBT are shown in Figure 4.13.

#### 4.4.2.2. Online HME recognizer

The overview of MTAP is shown in Figure 4.12. It includes three main parts: the feature extraction, the encoder, and the decoder.

##### A. Trajectory feature extraction

We utilized the set of point-based features used in [27]. An OHME is a sequence of trajectory points of pen-tip movements. We denote the sequence of  $L$  points as  $\{X_1, X_2, X_3, \dots, X_L\}$  with  $X_i = (x_i, y_i, s_i)$  where  $(x_i, y_i)$  are the coordination of each point and  $s_i$  is the corresponding stroke index of the  $i$ -th point. We store the sequence  $\{X_i\}$  in the order of writing process. Before extracting the features, we firstly interpolate and normalize the original coordinates according to [75]. They are necessary to deal with non-uniform sampling in terms of writing speed and the size variations of the coordinate by using different devices to collect patterns. For each point, we extract an 8-dimensional feature vector as follows:

$$[x_i, y_i, dx_i, dy_i, d'x_i, d'y_i, \delta(s_i = s_{i+1}), \delta(s_i \neq s_{i+1})] \quad (13)$$

where  $dx_i = x_{i+1} - x_i$ ,  $dy_i = y_{i+1} - y_i$ ,  $d'x_i = x_{i+2} - x_i$ ,  $d'y_i = y_{i+2} - y_i$  and  $\delta(\cdot) = 1$  when the conditional expression is true or otherwise zero, which presents the state of the pen (down/up).

##### B. Encoder

The encoder of MTAP is a combination of 4 stacked bidirectional GRUs [76] (BiGRUs) with a pooling operator, as shown in Figure 4.12. Stacking multiple BiGRU layers could make the model learn high-level representation from the input. The input

sequence of an upper BiGRU layer is the sequence of the hidden state of its lower BiGRU layer. Each layer has 250 forward and 250 backward units. Since the encoded features of two adjacent points are slightly different, the pooling layers are applied to reduce the complexity of the model and make the decoder part easier to parse with a fewer number of hidden states of the encoder. The pooling operator applied on the 2 top BiGRUs layers is to drop the even time steps of the lower BiGRU layer outputs and receive the odd outputs as the inputs. The hidden states outputted from the 4<sup>th</sup> layer are inputted to the decoder.

##### C. Decoder

The MTAP decoder receives the hidden states  $\{h_i\}_{i=1, \overline{K}}$  from the encoder and generates a corresponding LaTeX sequence of the input traces. The decoder consists of a word embedding layer of 256 dimensions, two layers of unidirectional GRU with 256 forward units, spatial attention, and a coverage model. The spatial attention points out the suitable

local region in  $\{h_i\}$  to attend for generating the next LaTeX symbol by assigning higher weights to a corresponding local annotation vector  $\{a_i\}_{i=1, \overline{K}}$ . The coverage model is a 1-dimensional convolutional layer to indicate whether a local region in  $\{h_i\}$  has been attended to the generation. The model is trained with an attention guider using the oracle alignment information from the training OHMEs to force the attention mechanism to learn well.

#### D. Training and testing

Table 4.8. ExpRate and CER of MTAP.

Dataset	MTAP		TAP [15]	
	ExpRate (%)	CER (%)	ExpRate (%)	CER (%)
CROHME 2014 testing set	48.88	14.54	50.41	13.39
Dset_Mix	34.62	17.51	-	-
NIER_CBT	57.89	18.57	-	-

We trained MTAP by the training data of CROHME 2016 on an Intel Xeon CPU@2.10GHz, a Tesla K80 GPU with 12Gb of RAM workstation. We removed genuine OHME patterns in Dset\_Mix for fair evaluation, because they are in the training data set. The optimizer and hyperparameters are the same as in [27]. Then, we measured the expression rate (ExpRate) and the character error rate (CER) on the testing data of CROHME 2014, Dset\_Mix, and NIER\_CBT, as shown in Table 4.8. The recognition rate of MTAP is 1.53 percentage points lower than the original TAP model on the CROHME 2014 testing set.

##### 4.4.2.3. Experiment settings

We utilized the  $k$ -means algorithm and the complete linkage (CL) method for the clustering task. For  $k$ -means, we applied the Euclidean distance and initialized centroids using  $k$ -means++ [69], a popular variant of the  $k$ -means algorithm that tries to spread out initial centroids. To evaluate the proposed features, we set the number of clusters as the number of categories in our experiments.

##### 4.4.2.4. Evaluation

In this section, we compare the proposed method with the previous methods. Moreover, we conduct experiments to evaluate our proposed SF.

#### A. Comparison with other methods

So far, clustering OHMEs is not shown on the common dataset. Here, we compare with our first approach presented in section 4.2.1. In addition, we compared with the method using the edit distance, which is to compute the dissimilarity between two LaTeX

Table 4.9. Comparisons with other methods of clustering HMEs. Values are presented in form of “average value (standard deviation)”

HME type	Name	Features	Clustering algorithm	Dset_Mix		NIER_CBT	
				Purity	MC	Purity	MC
OHME	Ours ( <i>M1</i> )	Online bag-of-features + DbR	<i>k</i> -means	0.777 (0.17)	0.629 (0.09)	0.898 (0.05)	0.702 (0.06)
	Ours ( <i>M2</i> )	MTAP + Edit distance + SbR	<i>k</i> -means	0.638 (0.12)	0.700 (0.05)	0.867 (0.05)	0.725 (0.07)
	Ours ( <i>M3</i> )	MTAP + GSSF	CL (GSSF)	0.806 (0.10)	0.611 (0.05)	<b>0.921</b> <b>(0.05)</b>	<b>0.698</b> <b>(0.06)</b>
	Ours ( <i>M4</i> )	MTAP + GSSF + SbR	CL (Euclidean)	0.775 (0.15)	0.633 (0.07)	0.920 (0.04)	0.700 (0.06)
	<b>Ours (<i>M5</i>)</b>	<b>MTAP + GSSF + SbR</b>	<i>k</i> -means	<b>0.916</b> <b>(0.05)</b>	<b>0.556</b> <b>(0.03)</b>	0.915 (0.03)	0.702 (0.07)
OfHME	Khuong et al. ( <i>M6</i> )	Offline bag-of-features	<i>k</i> -means	0.841 (0.15)	0.595 (0.07)	0.834 (0.07)	0.739 (0.08)
	Nguyen et al. ( <i>M7</i> )	CNN-based features	<i>k</i> -means	0.723 (0.16)	0.653 (0.08)	0.829 (0.06)	0.744 (0.06)

sequences outputted from MTAP, denoted as *M2*. SbRs produced by the edit distance are inputted to the *k*-means algorithm.

We carried out several experiments to evaluate the effectiveness of GSSF and SbR on the representation. Firstly, we directly used the absolute of GSSF as the distance function to input into CL, denoted as *M3*. Secondly, we used the SbR matrix produced by MTAP and GSSF to input into CL by using the Euclidean distance, denoted as *M4*. Thirdly, we used the SbR matrix produced by MTAP and GSSF to input into the *k*-means algorithm, denoted as *M5*.

We also compared our proposed method with previous methods for clustering offline HMEs (OfHMEs), which consists of the offline bag-of-features proposed by Khuong et al. [13] (denoted as *M6*) and the CNN-based features proposed by Nguyen et al. [14] (denoted as *M7*). OfHMEs are converted from OHMEs. We used a symbol classifier to extract the offline bag-of-features in *M6*. We also trained a CNN model to extract spatial classification features for *M7*. Those models in *M1*, *M6*, and *M7* were trained in the same dataset with MTAP.

Table 4.9 shows that our proposed GSSF combined with MTAP, i.e., *M3* and *M5*, outperforms *M1*, *M2*, and *M7* in purity and *MC* on both Dset\_Mix and NIER\_CBT. *M4* has slightly lower performance than *M1* on Dset\_Mix but it seems to be comparable. Also, *M3* and *M4* have lower performance than *M6* on Dset\_Mix. *M5* yields the best performance on Dset\_Mix, while *M3* performs best on NIER\_CBT. However, *M5* achieves a high purity on both datasets. Regarding *MC*, *M5* achieves the marking cost of around 0.556 and 0.702 in Dset\_Mix and NIER\_CBT. Consequently, the marking cost is reduced by 0.444 and 0.298 than manual marking.

There are some discussions based on the results of  $M3$ ,  $M4$ , and  $M5$ . Firstly, our GSSF without SbR works well when using the CL method on NIER\_CBT. Secondly, GSSF combined with SbR achieves more stable performance when using the  $k$ -means algorithm than the CL method with the Euclidean distance.

### B. Evaluations on our similarity function

We conducted experiments on forming our proposed GSSF. According to Eq. (7), GSSF is formed by taking an average of two similarity components  $F(S_1|S_2)$  and  $F(S_2|S_1)$  since we aim to make SF symmetric. We compare GSSF with three possible variants as follows:

- We directly use function  $F(x|y)$  as SF so that SbR of  $X_i$  in Eq. (8) is as  $[F(X_i|X_1), \dots, F(X_i|X_i), \dots, F(X_i|X_N)]$ . Since  $F$  is not a symmetric function, we denote this SF as Asymmetric\_GSSF.
- We define two SFs by getting the minimum and maximum value between  $F(S_1|S_2)$  and  $F(S_2|S_1)$  instead of taking the average of them. We denote them as Min\_GSSF

Table 4.10. Comparisons with other variants of SFs.

Method	Purity	
	Dset_Mix	NIER_CBT
Asymmetric_GSSF	0.857±0.07	0.907±0.05
Min_GSSF	0.861±0.08	0.907±0.04
Max_GSSF	0.909±0.08	0.913±0.04
<b>GSSF</b>	<b>0.916±0.05</b>	<b>0.915±0.03</b>

and Max\_GSSF, respectively.

Table 4.10 presents the performance of our proposed SF with Asymmetric\_GSSF, Min\_GSSF, and Max\_GSSF in terms of purity by using the  $k$ -means algorithm. Our GSSF performs better than the other SFs on Dset\_Mix and NIER\_CBT, which implies that taking the average of two similarity components is better than using them directly or taking the minimum or maximum value between them. However, Max\_GSSF yields comparable results with GSSF.

#### 4.4.2.5. Visualizing similarity-based representation matrix

This section shows the visualization of the SbR matrix to see how this representation discriminates for clustering. Figure 4.14 presents the SbR matrix of the subgroup 3 and 8 in Dset\_Mix. OHMEs belonging to the same class are placed together in both dimensions. For subgroup 3, its categories are significantly distinct. We can see that SbR well represents for OHMEs in the same category. The similarity scores among intra-category OHMEs almost near 0, and they are much higher than those among inter-category OHMEs. On the other hand, some categories in subgroup 8 are slightly different, such as categories  $C_3$  and  $C_5$  or category  $C_4$  and  $C_6$ , and SbR among them is not so different. Purity on subgroups 3 and 8 are 0.984 and 0.832, respectively.

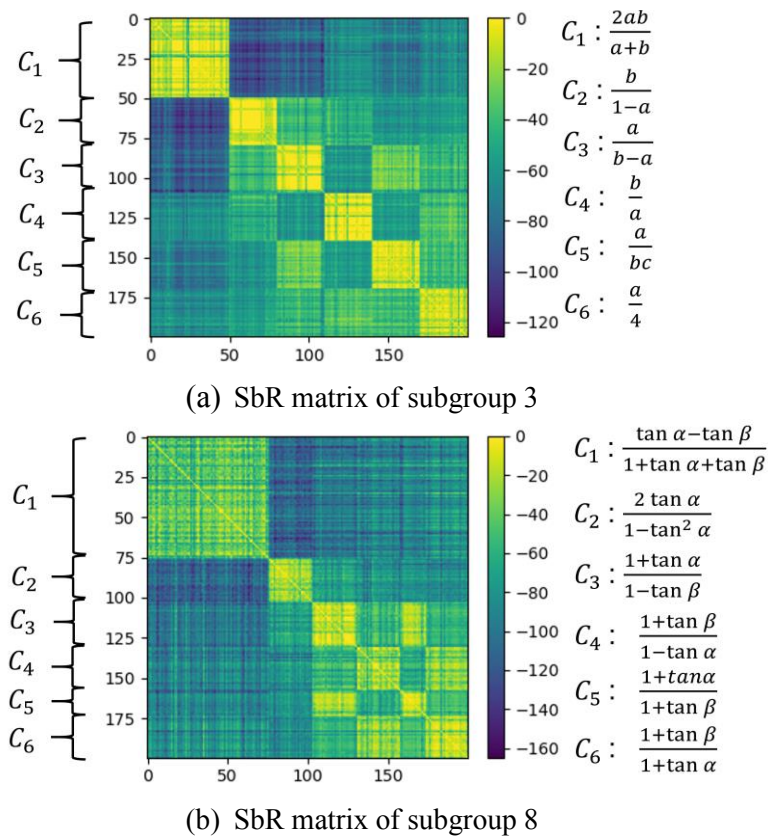


Figure 4.14. Visualization of the SbR matrix of the subgroup 3 and 8 before normalizing them into  $[0, 1]$ .

## 4.5. Conclusions

To provide computer-assisted marking for handwritten mathematics answers, this chapter presented two approaches for clustering OHMEs: (1) multi-level bag-of-features and (2) a generative sequence similarity function (GSSF) based on the Seq2Seq recognizer.

For the first approach, we presented multi-level bag-of-features consisting of a low-level type of image-based features from an OHME sample and high-level recognition-based symbolic and structural types of features obtained from an OHME recognizer. We

presented DbR to avoid the problem of applying the Euclidean distance on sparse feature spaces. We also proposed an approach to combine all types of features to enhance the performance of clustering OHMEs. We conducted experiments by using k-means++ on the Dset\_50 and Dset\_Mix datasets. On Dset\_50, which contains OHMEs with distinctly different math expressions, BoS with the DbR method alone achieved the highest performance without combining other types of features. Moreover, as the recognition accuracy of symbols and spatial relationships was higher, purity was improved. On Dset\_Mix, which is more similar to the realistic OHME answers, combining low-level features (Dir) and high-level features (BoS, BoR, PbBoS, and PbBoR) with DbR was better than using the individual types of features and some other combinations. For Dset\_Mix, setting the number of clusters according to the number of answers could be beneficial.

For the second approach, we presented a similarity-based representation (SbR) and a generative sequence similarity function (GSSF) based on the Seq2Seq recognizer for clustering to provide computer-assisted marking for handwritten mathematics answers OHMEs. The SbR matrix is then inputted to the  $k$ -means algorithm by setting the number of clusters as the number of categories. We achieved around 0.916 and 0.915 for purity and around 0.556 and 0.702 for the marking cost on the two answer datasets, Dset\_Mix and NIER, respectively. Our method outperforms other methods on clustering HMEs.



## CHAPTER 5. Online Handwritten Mathematical Expression Recognition

### 5.1. Introduction

There are challenging problems in OHME recognition. One problem is that there are lots of ambiguities in the interpretation of OHMEs. For instance, there exist math symbols that are very similar in the writing style, such as “0”, “o”, and “O” or dot and comma. These ambiguities challenge OHME recognition without utilizing contextual information. In this thesis, we propose two methods to address this problem.

The first method is to utilize bidirectional context from input stroke sequences for symbol segmentation and classification using deep Bidirectional Long-Short Term Memory (BLSTM) encoder. Discriminating ambiguous symbols requires the adoption of the context from other strokes. Conventional methods sequentially implement symbol segmentation and symbol classification so that the classification step is made by only using the local context obtained from segmentation hypotheses. Therefore, the ambiguous symbols are challenging to be distinguished in the symbol classification step. In handwritten text recognition, temporal recognition of characters benefits from bidirectional context of preceding strokes and succeeding strokes by a Recurrent Neural Network (RNN) [77]. Instead of considering isolated character recognition, we consider recognition of the symbols in an HME as temporal recognition, where the bidirectional context from input stroke sequences is used for symbol segmentation and classification. The deep BLSTM processes an input HME as a stroke sequence to produce a sequence of classification probability corresponding to the input sequence. The segment hypotheses are produced by making queries to retrieve the symbol recognition probability. We also derive the method to detect junk symbols from the retrieved recognition probability without learning junk symbols.

The second method is to utilize a math language model combined with OHME recognizers. Here, we present the first transformer-based math language model (TMLM). Based on the self-attention mechanism, the high-level representation of an input token in a sequence of tokens is computed by how it is related to the previous tokens so that TMLM can capture long dependencies and correlations in MEs. Then, we propose a method to combine TMLM into a stochastic context-free grammar-based HME recognizer. In our experiments, we show that our TMLM outperforms the traditional  $N$ -gram model and RNNLM in the task of modeling MEs.

The rest of this chapter is organized as follows. Section 5.2 describes our proposed methods in detail. Section 5.3 presents our experiments for evaluating the proposed method. Finally, section 5.4 concludes our work.

## 5.2. Proposed methods

In this section, we firstly present a method for online handwritten math symbols segmentation and classification for improving OHME recognition. Then, we present a transformer-based math language model.

### 5.2.1. Online handwritten mathematical symbol segmentation and recognition with bidirectional context

We improve the symbol recognition of HME by using a deep BLSTM-CTC model to encode global context information.

#### 5.2.1.1. Bidirectional context for symbol classification

We improve the context for symbol classification by considering the classifier, which incorporates the whole input sequence instead of individual handwritten symbols. An input sequence is processed by bidirectional recurrent neural networks where the classification of each time step can access the context from both forward and backward directions of input. The classification of symbols at every time step is used for retrieving the classification of segmentation hypotheses.

#### 5.2.1.2. Temporal classification with RNN

We apply a deep Bidirectional Long Short-Term Memory (BLSTM) to incorporate bidirectional context for symbol classification as shown in Figure 5.1. A BLSTM is a combination of two LSTM layers which process the input in forward and backward

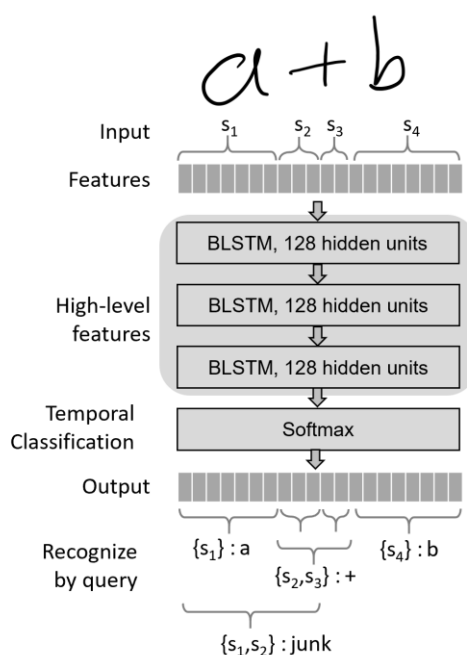


Figure 5.1. Symbol classification by strokes query.

directions [78]. The forward and backward context by the two LSTM layers is combined and feed to the next BLSTM layer in the networks. Deep BLSTM stack multi-level of BLSTM to learn high-level features. LSTM [79] is an advanced architecture of RNN designed to overcome the problem of vanishing or exploding gradients allow it to incorporate long-range context for improving handwriting recognition.

CTC [77] is an objective function for RNN designed to make RNN learn directly from an input sequence to a target sequence without requiring pre-segmented input. The alignment between the input sequence and output label sequence is learned automatically with the assumption that the two sequences are in the same order.

CTC introduces a label called ‘blank’ that denotes no label. It defines the output  $y^t$  of RNN for each time  $t$  with respect to an input sequence  $x$  length  $T$  as the probability distribution over a fixed set of classes  $C$  and the ‘blank’ label.

$$y_k^t = p(k, t | x), \forall k \in C \cup \text{blank} \quad (14)$$

where  $y_k^t$  is the output  $y^t$  for class  $k$ .

An output label sequence  $l$  is obtained by a reduction process  $B$  over a path  $\pi_{1:T} = k_1, k_2, \dots, k_T$  through the lattice of output labels, i.e.,  $k_i \in C \cup \text{blank}, i = \overline{1, T}$ . The reduction process firstly removes repeated labels, then removes ‘blank’ labels in this path.

$$p(l, \pi_{1:T} | x) = p(\pi_{1:T} | x) = \prod_{t=1}^T p(k_t, t | x) \quad (15)$$

where  $l = B(\pi)$ .

The probability for output label sequence  $l$  from an input sequence  $x$  is the total probability of all the paths  $\pi_{1:T}$ , where each path is reduced into  $l$ .

$$p(l | x) = \sum_{\pi_{1:T} \in B^{-1}(l)} p(l, \pi_{1:T} | x) \quad (16)$$

where  $B^{-1}(l)$  is the set of all paths which reduced into  $l$ .  $p(l | x)$  is calculated by CTC forward-backward algorithm applied to the temporal classification output  $y$  [77].

For a pair of input sequence  $x$  and output sequence  $l$  from the training dataset, the network is trained by minimizing CTC loss obtained by:

$$\text{Loss} = -\log(p(l | x)) \quad (17)$$

### 5.2.1.3. Symbol classification by strokes query

As for the traditional symbol classifier, a sequence of strokes is inputted to the recognizer. Instead of inputting the stroke sequence to the classifier, we use it to retrieve the classification results from temporal classification.

We first obtain the temporal classification output  $y$  for an input sequence of  $n$  strokes  $x = (s_1, s_2, \dots, s_n)$  by deep BLSTM. Let  $r_1, r_2, \dots, r_n$  denote the range of time steps corresponding to  $s_1, s_2, \dots, s_n$ , respectively. For a hypothesis  $h = (s_i, \dots, s_j) \subset (s_1, s_2, \dots, s_n)$  we obtain the query output  $y' = y^{(r_i:r_j)}$  as the output  $y$  in range of time steps corresponding to hypothesis  $h$ .

The probability of a symbol hypothesis  $h$  belonging to class  $c$  derived from (16) is described as follows:

$$p(c|h) = \sum_{\pi' \in B^{-1}(c)} p(c, \pi'|x) \quad (18)$$

where  $\pi'$  is a path through the time steps of the hypothesis  $h$  produces the  $c$ -class symbol.

The probability  $p(c|h)$  is calculated by the CTC forward-backward algorithm on the query output  $y'$  as similar to  $p(l|x)$ . However, applying the algorithm for all the symbol classes in each recognition is not practical due to its complexity. For implementation, we make a fast approximation of (18) as the probability of the highest path to produce  $c$ -class symbol:

$$p(c|h) \approx \max_{\pi' \in B^{-1}(c)} p(c, \pi'|x) \quad (19)$$

The best path to produce  $c$ -class symbol would have the form of  $\hat{\pi}' = \{(\text{blank})_u(c)_v(\text{blank})_w\}$ ,  $u, w \geq 0, v \geq 1$ , where  $(symbol)_n$  denotes the repeat of the symbol  $n$  times. We approximate the probability of outputting  $c$ -class through  $\hat{\pi}'$  by

$$p(c, \hat{\pi}'|x) = \prod_{t; \hat{\pi}'_t = \text{blank}} y'^t_{\text{blank}} \times \prod_{t; \hat{\pi}'_t = c} y'^t_c \quad (20)$$

where  $t$  is the time step of the query output  $y'$ ,  $y'^t_k$  denotes the probability of class  $k$  at the time step  $t$ .

From Eq. (19) and Eq. (20) we obtain the approximation for symbol classification  $p(c|h)$  on the query output  $y'$ . The approximation in Eq. (20), however, requires finding the best

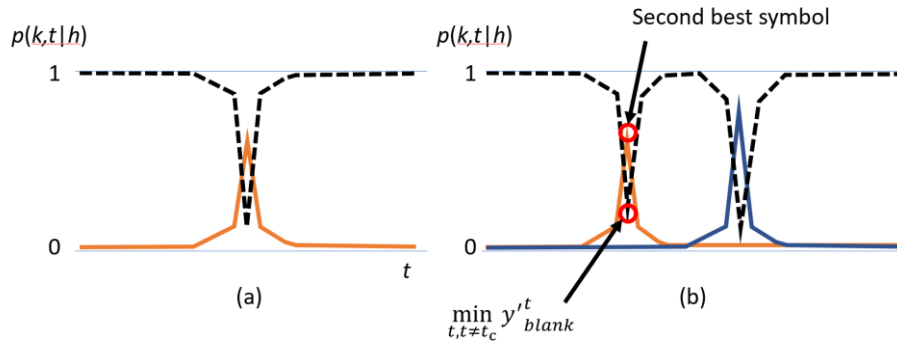


Figure 5.2. Temporal classification probability: (a) single symbol (b) two symbols. Dashed line shows the probability of 'blank' symbol.

path  $\widehat{\pi}^t$  for all the class  $c$  in each symbol recognition. We make it simpler by considering  $\prod_{t; \widehat{\pi}^t=c} y'^t_c$  as the symbol recognition probability and  $\prod_{t; \widehat{\pi}^t=blank} y'^t_{blank}$  as the penalty if the hypotheses contain more than one symbol. We visualize symbol recognition probability for two cases of one-symbol hypotheses and two-symbols hypotheses in Figure 5.2. Typically, the output yields the number of peaks as the same as the number of symbols and the remaining are ‘blank’. For the case of one symbol,  $\prod_{t; \widehat{\pi}^t=c} y'^t_c$  is at the peak and there is no penalty since  $y'^t_{blank}$  is equal to 1.0 at the base. For the case of two or more symbols,  $\prod_{t; \widehat{\pi}^t=c} y'^t_c$  is at the highest peak and the penalty could be determined by  $\min_{t, \widehat{\pi}^t=blank} y'^t_{blank}$ , which is the point of the second-best peak.

Let  $\widehat{c}$  be the symbol at the second-best peak and  $n_s$  is the number of recognized symbols in the hypotheses, we obtain the symbol probability as follows:

$$p(c|h) \approx \max_t y'^t_c \times \left(1 - \max_t y'^t_{\widehat{c}}\right)^{n_s-1} \quad (21)$$

where the penalty  $\left(1 - \max_t y'^t_{\widehat{c}}\right)^{n_s-1}$  is active when there is more than one symbol in the hypotheses.

For implementation, we obtain  $n_s$  by making the best path decoding method [77] and find  $\widehat{c}$  by getting the second-best symbol of the probability  $p(c|h) = \max_t y'^t_c$ . There are cases that  $\widehat{c}$  is not obtainable by  $p(c|h)$  due to it is the same with the best symbol. Therefore, we further add a penalty of 0.5 to deal with the problem since a peak is typically larger than 0.5. We obtain the formula as in (9):

$$p(c|h) \approx \max_t y'^t_c \times \left(1 - \max_t y'^t_{\widehat{c}}\right)^{n_s-1} (0.5)^{n_s-1} \quad (22)$$

The method is also appropriate for delayed handwritten strokes, where the strokes of the current symbol are written after the strokes of other symbols.

For stroke query classification, the alignment must be correct at stroke level. The requirements are in the formula:

$$Loss = - \sum \log(p(c|h)) \quad (23)$$

where the pair  $(h, c)$  is obtained from the ground truth of stroke label alignment.

We may use the alignment loss with stroke symbol annotations for providing correct alignment for the networks to learn. As in experiments, we found that the model could learn proper stroke alignment without supervised alignment annotations. The alignment could be learned correctly as a similar mechanism of weakly supervised learning for object detection by CNNs [80,81].

#### 5.2.1.4. Detect junk symbols

The probability of junk class is obtained from the penalty for the hypotheses containing more than one symbol as in Eq. (22):

$$p(junk|h) \approx 1 - \left(1 - \max_t y'_t \frac{t}{c}\right)^{n_s-1} (0.5)^{n_s-1} \quad (24)$$

We can also obtain it after the probability for all the symbols is calculated:

$$p(junk|h) = 1 - \sum_{c \in \mathcal{C}} p(c|h) \quad (25)$$

### 5.2.1.5. Online features

For online handwriting recognition, various features have been studied [82,83]. Some spatial features such as distance and differences ( $\Delta x$  and  $\Delta y$ ) between two adjacent coordinates, pen up/down information, curvature at each point, etc. Also, aspect and curliness of trajectory, stroke slope and linearity are applied as well. All of the above features are point-based features, i.e., they are extracted from each point of pen trajectory. Hence, they are known as local features.

In our experiment, input stroke sequences are firstly sampling the coordinates by Ramer methods [84]. For each sampled point, we then extract four basic features: the sine and cosine of the writing directions, the normalized distance between the preceding and the succeeding points of the current point, and a binary value of pen state (pen-up/pen-down).

### 5.2.2. Transformer-based math language model

Given a sequence of tokens  $X = (x_1, x_2, \dots, x_N)$ , constructing a language model is to estimate the joint probability  $P(X)$ , which is often auto-regressively factorized as  $P(X) = \prod_t P(x_t|X_{<t})$  where  $X_{<t} = (x_1, \dots, x_{t-1})$ . According to this factorization, the problem reduces to estimating each conditional factor  $P(x_t|X_{<t})$ . In this thesis, our proposed model with a self-attention mechanism encodes the context  $X_{<t}$  to produce the categorical probability of the token  $x_t$ .

In this section, we first describe our proposed TMLM, which is mainly based on [85]. Then, we present a method for combining our model with an HME recognizer.

#### 5.2.2.1. Proposed language model

TMLM consists of three main parts: an input embedding layer, a positional encoding layer (PE), and a stack of transformer layers, as shown in Figure 5.3. First, sequential input tokens  $\{x_1, x_2, \dots, x_N\}$  are fed into the input embedding to embed the categories of discrete tokens into a continuous space for better representation. Secondly, each embedded vector according to each input token is added by a PE vector to present the token's position in the sequence. The detail of the PE is presented later in this section. Thirdly, the outputs of the input embedding, and positional encoding are passed into

stacked transformer layers to learn high-level representation based on the self-attention mechanism. Finally, the output of the top transformer layer is input to a softmax layer to obtain the categorical probability for the token  $x_t$  given  $\{x_1, \dots, x_{t-1}\}$ . Although all input tokens are fed into our model at the same time, the model is restricted to attend only tokens on the left side of  $x_t$  to produce  $P(x_t|x_1, \dots, x_{t-1})$  by a mask in the transformer layer.

The architecture of the transformer layer is based on the decoder of the conventional transformer-based model [52]. It consists of a masked multi-head self-attention (MMSA), layer normalization [86], and a feedforward neural network, as shown in Figure 5.3. In addition, residual connections are added for the model to learn better. Here, we present MMSA and PE, which play important roles in our model.

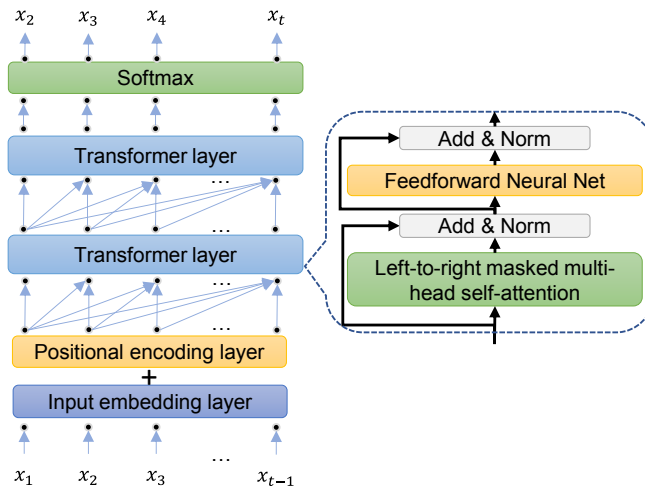


Figure 5.3. Overview of the proposed transformer-based language model with two transformer layers.

### A. Masked multi-head self-attention

This layer receives the representation of input tokens and outputs the higher representation for the tokens based on how each token is related to others. MMSA includes multiple attention functions, which allow the model to attend information from different representation subspace. We firstly present a masked single-head self-attention.

A traditional attention function can be described as the mapping of a query and a set of key-value pairs to produce an output. Note that the query, the keys, and the values are all vectors. The output is a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key of the value.

The masked single-head self-attention function, called scaled dot-product attention (SDPA), are also based on the queries ( $Q$ ), the keys ( $K$ ) of dimension  $d_k$ , and the values

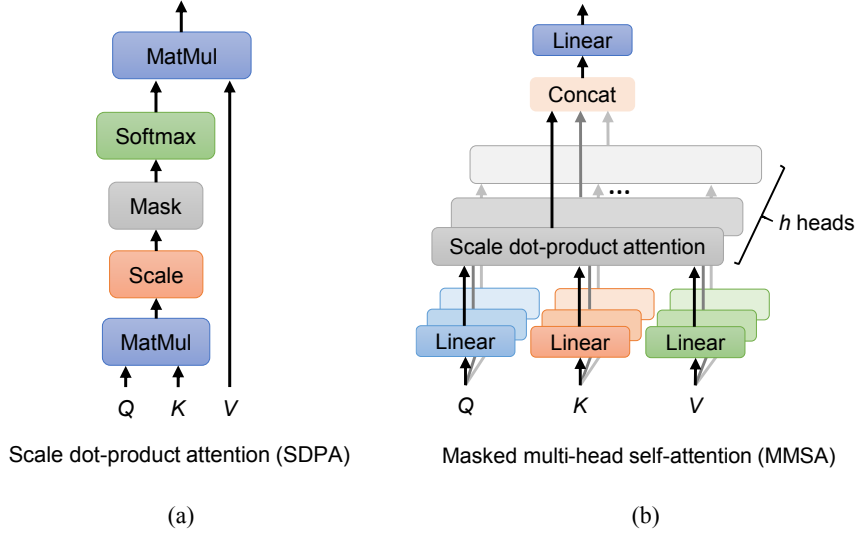


Figure 5.4. Illustration of scale dot-product attention and masked multi-head attention.

( $V$ ) of dimension  $d_v$ , as shown in Figure 5.4(a). We compute the dot products of the query with all keys, then scale them by  $\sqrt{d_k}$ . Next, we apply a mask to restrict the model to attend only the left side of the current predicted token. We then apply a softmax function to obtain the weights on the values. The output of this attention function is formulated as follows:

$$Att(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (26)$$

SDPA is called a “head” in MMSA. The architecture of MMSA including  $h$  heads is shown in Figure 5.4(b). With multiple heads, we project the queries, keys, and values  $h$  times with three different learnable linear projections. On each of these projected versions of queries, keys, and values, we then perform SDPAs in parallel. Then, we concatenate their outputs and once again project to obtain the final output of MMSA.

## B. Positional encoding

Since tokens  $(x_1, x_2, \dots, x_N)$  are input to our model at the same time and there is no convolutional/recurrent layer, the model cannot exploit the positional information of tokens. It is a serious problem for the task of language modeling. To address it, we utilize PE having the same dimensionality as the input embedded vector,  $R^{N \times d_{embed}}$  ( $d_{embed}$  is the dimension of the input embedded vector). Then, we add PE to the input embedded vector to provide the positional information for our model. PE of the  $p$ -th token and the  $i$ -th dimension is computed by the sine and cosine function as follows:

$$PE(p, i) = \begin{cases} \sin\left(\frac{p}{10000^{i/d_{embed}}}\right) & \text{if } i \text{ is even} \\ \cos\left(\frac{p}{10000^{(i-1)/d_{embed}}}\right) & \text{otherwise} \end{cases} \quad (27)$$



### 5.2.2.2. Combining language model with HME recognizer

In this study, we use a language model to sort the top- $M$  best candidates outputted from the stochastic context-free grammar-based HME recognizer. Given  $M$  candidates  $\{c_1, c_2, \dots, c_M\}$  of LaTeX sequences and their corresponding scores, the combined scores are computed as follows:

$$Score_{comb}(c_i) = Score_{recog}(c_i) + \alpha \times Score_{LM}(c_i) \quad (28)$$

where  $Score_{recog}(c_i)$  and  $Score_{LM}(c_i)$  are the scores of the  $i$ -th candidate,  $c_i$ , from the HME recognizer and the language model, respectively.  $Score_{comb}(c_i)$  is the combined score of  $c_i$ .  $\alpha$  is a weighting parameter to balance between recognition and language scores. Note that  $Score_{LM}(c_i)$  is the sum of logarithms of conditional probabilities output from the language model and normalized by the length of the candidate,  $c_i$ . For this combination method, we refer to the HME recognizer producing  $Score_{recog}(c_i)$  based on the sum of logarithms of probability terms. The candidate having the highest combined score is the final recognition result.

## 5.3. Experiments

In this section, we present evaluations of our proposed method for improving OHME recognition on the CROHME datasets.

### 5.3.1. Experiments on online handwritten mathematical symbol segmentation and recognition

In this section, we conducted the experiments to assess the performance of context for mathematical symbol classification and the overall performance for ME recognition. The deep BLSTM was trained using the Tensorflow library. The experiment of the recognition system was run on an Intel Core i9 9900X CPU@3.5 GHz desktop PC.

#### 5.3.1.1. Dataset

Table 5.1. CROHME 2016 dataset

	Train	Validation	Test
# Symbols	85802	10061	10019
# Junk symbols	-	-	8416
# Expressions	8836	986	1147

We conducted the experiments on CROHME 2016 competitions [67] for both the isolated symbol recognition and mathematical expression recognition. The detail of the datasets used for training, validation and testing is shown in Table 5.1. The number of symbol's classes is 101.

We extracted a supervised label sequence of symbols for each mathematical expression from the stroke-level annotations in the dataset. The label sequence is represented in the same order with the writing order of strokes.

### 5.3.1.2. Experimental settings

We use a stack of three BLSTM layers. Each BLSTM contains two LSTM layers with 128 cells. The outputs of each time step by two LSTM layers are concatenated into a feature vector of 256 dimensions before input into the next BLSTM layer. We trained the networks by Stochastic Gradient Descent (SGD) with a learning rate of 0.0001 and a momentum of 0.9.

For evaluation, we evaluated the symbol classification by stroke query in temporal output and evaluated the expression rate in recognizing HME. To evaluate junk symbol detection, we used the same evaluation method of false acceptance rate (FAR) and true acceptance rate (TAR) as in [67]. Here, accepted symbols or accepted ‘junk’ means the symbols or ‘junk’ are classified as valid symbols by the system.

$$\text{TAR} = \frac{\# \{\text{accepted symbols}\}}{\# \{\text{total symbols}\}} \quad (29)$$

$$\text{FAR} = \frac{\# \{\text{accepted junk}\}}{\# \{\text{total junk}\}} \quad (30)$$

### 5.3.1.3. Results

We show the results of symbol classification with context and without context in Table 5.2. The model with context is named BLSTM\_CTC and that without context is named BLSTM\_iso. The architecture of BLSTM\_iso is also composed of three BLSTM layers of 128 cells each. With bidirectional context, the model better learns symbol classification due to the reduction of ambiguous symbols. Consequently, symbol classification results in test set improve from 89.55% to 92.30% as a reduction of 26.32% of the classification error. The recognition rate almost comparable with MyScript and Tokyo, although MyScript used additional training sets and Tokyo used a combination of online and offline recognizers. For junk symbol detection, BLSTM\_CTC got the best TAR of 98.03%,

Table 5.2. Symbol classification in CROHME 2016.

	<b>101</b>	<b>102</b>	<b>TAR</b>	<b>FAR</b>
	<b>classes</b>	<b>classes</b>		
<b>MyScript</b>	92.81	86.77	89.82	11.16
<b>Tokyo</b> [94]	92.27	-	-	-
<b>RIT</b> [95]	88.85	83.34	95.86	19.71
<b>BLSTM_CTC</b>	92.30	84.82	<b>98.03</b>	23.24
<b>BLSTM_iso</b>	89.5	50.79	84.86	89.76

which is important since it rejects fewer symbols. Without training on junk symbols, BLSTM\_iso is unable to reject junk symbols as FAR being 89.76%.

We notice the improvements in recognition of ambiguous symbols that are difficult to discriminate without context. Table 5.3 shows the differences of recognition rate by symbol class by the recognition system with and without context. The symbols of \prime, O, V, C, X which are unable to be recognized by the system without context, can be recognized by the system with context.

Table 5.3. Class-based recognition comparison.

	<b>Symbols</b>	<b>No context</b>		<b>With context</b>	
		<b>Correct</b>	<b>Most confusion</b>	<b>Correct</b>	<b>Most confusion</b>
<b>Improved</b>	\prime	0.00	l (0.55)	0.38	, (0.23)
	O	0.00	o (0.91)	0.33	o (0.67)
	V	0.00	v (0.73)	0.33	v (0.60)
	C	0.06	c (0.87)	0.47	c (0.36)
	X	0.07	x (0.80)	0.41	x (0.46)
	.	0.19	- (0.33)	0.89	1 (0.05)
	s	0.24	S (0.67)	0.43	5 (0.23)
	,	0.34	) (0.22)	0.85	1 (0.14)
<b>Degraded</b>	\lambda	0.43	h (0.29)	0.00	) (0.40)
	\div	0.89	= (0.11)	0.11	+ (0.39)
	Y	0.15	y (0.77)	0.12	y (0.88)
	\sigma	0.54	\theta (0.38)	0.13	\infty (0.25)
	\mu	0.71	u (0.14)	0.14	u (0.86)

Although symbol recognition has been improved by using the context, the recognition rate of almost ambiguous symbols is still around 40%. Moreover, there are some symbols which their recognition rate is decreased when applying the context. This denotes that there is still a room to encode better context or combine the methods using and without using context to enhance the performance of recognizing the symbols.

We verify the effect of the improvement to the performance of the HMEs recognition. We combine the symbol recognition with an SCFG based parser to build an online HMEs

recognition module. The system with BLSTM\_iso uses the segmentation provided by Le et al. [87]. We then evaluate the module on CROHME 2016 and show the results in Table IV. From the results, the recognition rate of HMEs by using the CTC based symbol recognition is 44.81%, which is nearly 17 points higher than that of 27.72% by the model using the BLSTM based symbol recognition. Moreover, the recognition rate with the number of errors less or equal than 1 and 2 are also increase to 57.02% and 60.94%, which are higher than those recognition results of the module using BLSTM with additional junk pruning provided by BLSTM\_CTC. The results show that the better symbol recognizer helps to recognize HMEs more correctly.

We also compare the HMEs recognition module with other state-of-the-art methods on CROHME 2016 and show the results in Table 5.4. Details of other works can be found in the CROHME competition [67]. We can see that our method using BLSTM\_CTC symbol recognizer accounts for a competitive result compared with other methods.

Recognizing isolated characters one-by-one incurs the symbol recognition complexity of  $o(n^2)$  for an  $n$ -strokes expression while recognizing by CTC only performs once. Symbol query does not need much calculation; therefore, the recognition system largely speeds up. Theoretically, recognizing by stroke query incurs symbol recognition complexity of  $o(1)$  for an HME.

Table 5.4. Expression rate (%) compare with state-of-the-arts on CROHME 2016.

<b>System</b>	<b>Expr. Rate (%)</b>	<b>&lt;= 1 error</b>	<b>&lt;= 2 errors</b>
<b>Wiris</b>	49.61	60.42	64.69
<b>Tokyo [87]</b>	43.94	50.91	53.70
<b>Sao Paolo [88]</b>	33.39	43.5	49.17
<b>Nantes</b>	13.34	21.02	28.33
<b>TAP [27]</b>	57.02	72.28	75.59
<b>BLSTM_iso</b>	16.56	20.66	22.84
<b>BLSTM_iso+JunkPrun</b>	27.72	37.31	42.28
<b>BLSTM_CTC</b>	44.81	57.02	60.94

We show the recognition time of the two systems with pruning by segmentation and pruning by junk symbol detection in Figure 5.5. Junk symbol detection makes slightly better than rejection by segmentation classifier. Using a CTC classifier with a stroke query clearly reduces the recognition time even for ME with a large number of strokes.

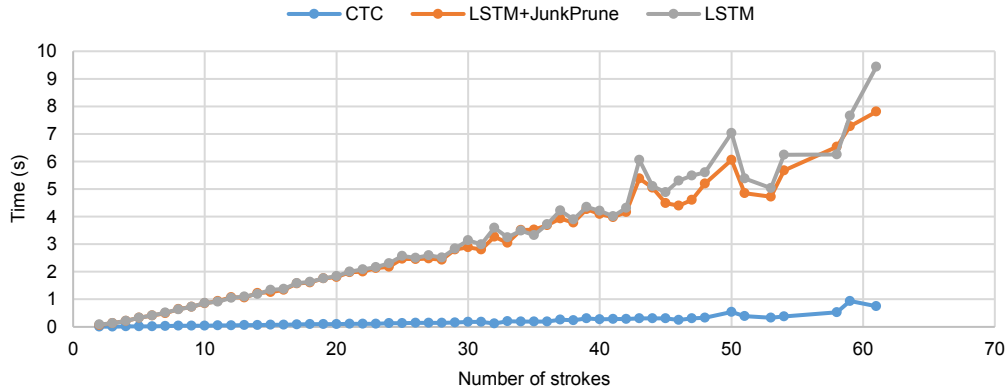


Figure 5.5. HME recognition time according to the number of strokes.

### 5.3.2. Experiments on transformer-based math language model

In this section, we present evaluations for our proposed TMLM on CROHME datasets.

#### 5.3.2.1. Dataset

We use a corpus of 68,862 LaTeX sequences provided in CROHME 2016 [67]. For preprocessing steps, we first filtered invalid syntax LaTeX sequences and removed style-related characters such as “ $\backslash\mathrm{}$ ”, “ $\backslash\mathrm{trm}$ ”, and so on. Then, we normalized the LaTeX sequences into the same format as the output of our HME recognizer. For example, “ $\{a\}^{\{2\}}$ ” is normalized as “ $a^{\{2\}}$ ”. The corpus is partitioned into a training set, a validation set, and a testing set according to the ratio of 8:1:1. The number of symbols in the dictionary is 108, including the padding “ $\langle\mathrm{pad}\rangle$ ” and the end-of-sequence symbol “ $\langle\mathrm{eos}\rangle$ ”.

#### 5.3.2.2. HME recognizer

In this section, we present the online HME recognizer [89] used in our experiments. The recognizer receives a sequence of point-based features extracted from an input HME and outputs recognition result as a LaTeX sequence. It consists of two main stages: (1) A symbol-relation temporal classifier (SRTC) for segmenting and classifying symbols and spatial relationships in an HME and (2) A symbol-level parser (SLP). We denoted this HME recognizer as SRTC\_SLP.

##### A. Symbol-relation temporal classifier

SRTC consists of three stacked Bidirectional Long-Short Term Memory (BLSTM) layers and a Connectionist Temporal Classification (CTC) layer at the top, as shown in Figure 5.6. Its input is a sequence of point-based features, including the representation of off-strokes (pen movements between strokes). Here, we ideally assume that there are no delayed strokes in the input HMEs. The stacked multiple BLSTM layers encode bidirectional context from the input and learn high-level representation. Then, the CTC layer generates a sequence of symbols and spatial relationships. There are 7 types of spatial relationships: superscript, subscript, upper, lower, horizontal, inside, and no relation (denoted as “NoRel”).

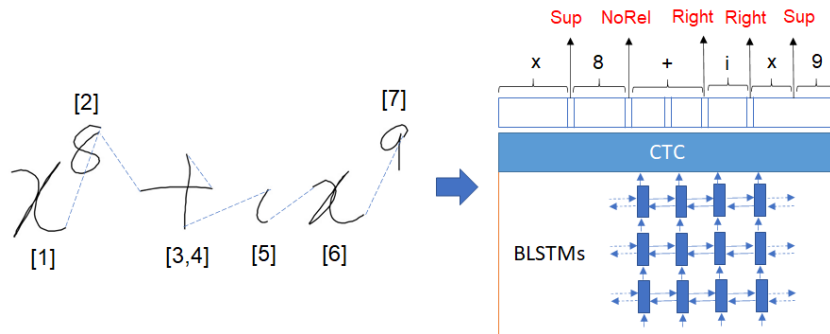


Figure 5.6. Illustration for symbol-relation temporal classifier.

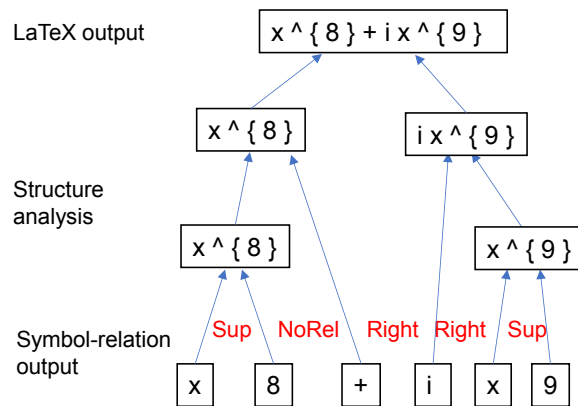


Figure 5.7. Symbol-level parser.

## B. Symbol-level parser

Given the output of SRTC, SLP based on the Cocke–Younger–Kasami (CYK) algorithm [90] is applied to merge recognized symbols and spatial relationships along with predefined grammar rules, as shown in Figure 5.7. This bottom-up method considers many possible combinations of hypotheses at the intermediate levels. Hence, it produces several candidates at the top of the combination tree even if the less promising candidates are pruned. Each candidate has a corresponding score computed based on the classification probabilities of symbols and spatial relationships.

### C. Training and testing

SRTC\_SLP was trained on the CROHME 2016 and CROHME 2019 training sets and tested on the CROHME 2016 and CROHME 2019 testing sets, respectively. Without using a language model, it achieved the expression rate of 53.44% and 52.38% on the CROHME 2016 and CROHME 2019 testing sets, respectively. This expression rate is higher than that of the state-of-the-art TAP recognizer (without using language models and/or ensemble methods) [27] by 3.22 percentage points in the expression rate on the CROHME 2016 testing set.

#### 5.3.2.3. Experimental settings

In this section, we present settings for training the proposed TMLM. We run our experiments on an AMD Ryzen 32 core processor CPU@2.16GHz, an RTX A6000 GPU with 48Gb RAM workstation. We evaluated TMLM with different numbers of transformer layers. The number of heads is fixed to 4 heads. The dimension of each head is set to 16. The context length of TMLM is fixed to 256, which covers the maximum sequence length in our LaTeX corpus. The dimension of vectors of input embedding and the dimension of hidden states are set to 256 and 512, respectively. The number of hidden nodes in the feedforward neural net layer is set to 1024. The dropout rate is set to 0.1. We applied an adaptive log-softmax function proposed in [91]. Our model is trained by the AdamW optimizer [92] with a learning rate of  $10^{-5}$ . The model is implemented based on the “Hugging Faces” library [93]. For combining language models with SRTC\_SLP, we determined the parameter  $\alpha \in R^+$  in Eq. (28) by applying the enumeration method on  $\{0, 0.1, 0.2, \dots, 2.0\}$ . The chosen  $\alpha$  parameters achieved the best expression rates on the CROHME 2014 testing set.

To evaluate language models, we utilized the perplexity measurement. Given a sequence of tokens  $X = (x_1, x_2, \dots, x_N)$ , the perplexity of  $X$  is the exponentiated average negative log-likelihood formulated as follows:

$$ppl(X) = \exp \left\{ -\frac{1}{N} \sum_{t=1}^N \log p(x_t | X_{<t}) \right\} \quad (31)$$

where  $p(x_t | X_{<t})$  is the conditional probability outputted from the language model.

#### 5.3.2.4. Evaluation

In this section, we compare the proposed language model with the previous methods. Then, we conduct experiments to compare the performance of those models when combined with the SRTC\_SLP recognizer.

### A. Comparisons with other language modeling methods

We compared our TMLM with the traditional  $N$ -gram model and GRULM. We increased the context length  $N$  in the  $N$ -gram model to 11 since TMLM can attend all the past contexts for a fair comparison. For GRULM, we increased the number of GRU layers up to 3 layers for evaluating the performance as well as comparing with TMLM in the condition of a similar number of trainable parameters. The dimension of an input embedded vector and hidden states in GRULM are set as the same as in TMLM.

Table 5.5. Comparisons with other language modeling methods.

<b>Model</b>	<b>#Layers in model</b>	<b>#Parameters</b>	<b>Perplexity</b>
3-grams	-	-	9.603
5-grams	-	-	7.557
9-grams	-	-	6.550
11-grams	-	-	6.500
<b>GRULM_1L</b>	1	1.3M	6.050
<b>GRULM_2L</b>	2	2.8M	6.049
<b>GRULM_3L</b>	3	4.4M	6.377
<b>Ours: TMLM_2L</b>	2	2.7M	4.598
<b>Ours: TMLM_5L</b>	5	6.3M	4.509
<b>Ours: TMLM_8L</b>	<b>8</b>	<b>10M</b>	<b>4.420</b>

Table 5.5 presents the perplexity of the models on the testing set extracted from our LaTeX corpus, as mentioned in section 4.3. The results show that our proposed TMLM models outperform all N-gram models and all GRULMs, even using fewer trainable parameters. For the N-gram models, increasing the context length can improve the perplexity, but it seems to converge when N reaches 11. GRULMs perform better than the N-gram models. Among GRULMs, the perplexity of GRULM\_2L achieves the best, which implies that increasing the number of GRU layers is not adequate. On the other hand, TMLMs can learn better when increasing the number of transformer layers.

With nearly the same number of trainable parameters, TMLM\_2L performs significantly better than GRULM\_2L. It implies that the architecture of TMLM is much more effective than the traditional GRULM on modeling MEs.

### **B. Evaluation on combining language models into the HME recognizer**

We combined the SRTC\_SLP recognizer with the language models that achieved the best performance in the previous experiment (i.e., 11-grams, GRULM\_2L, and TMLM\_8L). In detail, the combined score in Eq. (28) is computed for the top-10 best candidates from SRTC\_SLP.



Table 5.6 presents the expression rates of the combined recognizers on the CROHME 2016 and CROHME 2019 testing sets. The combination of SRTC\_SLP and TMLM\_8L achieves the best expression rates in both testing sets. TMLM\_8L improves 2.97 and 0.83 percentage points of the expression rates on the CROHME 2016 and CROHME 2019 testing set, respectively. The SRTC\_SLP + 11-grams is better than SRTC\_SLP + GRULM\_2L in the CROHME 2016 testing set, but that result is opposed in the CROHME 2019 testing set.

Table 5.6. Expression rates on combining the HME recognizers with language models.

Recognition system	Expression rate (%)	
	CROHME 2016	CROHME 2019
SRTC_SLP	53.44	52.38
SRTC_SLP + 11-grams	56.15	52.54
SRTC_SLP + GRULM_2L	55.36	52.88
<b>(Ours) SRTC_SLP + TMLM_8L</b>	<b>56.41</b>	<b>53.21</b>
(Zhang et al. [27]) TAP	49.29	-
(Zhang et al. [27]) TAP + GRUs	50.41	-
(Wu et al. [46]) PAL-v2	49.00	-
(Wu et al. [46]) PAL-v2 + 4-grams	49.35	-

LM: language model

Table 5.6 also presents the expression rates of the state-of-the-art HME recognizers that utilized math language models, i.e., TAP [27] and PAL\_v2 [45]. Compared to those models, SRTC\_SLP combined with our TMLM\_8L yields the best expression rate on the

Table 5.7. Percentages of corrected, miscorrected, and unchanged recognition results when combining the SRTC\_SLP recognizer with language models.

Dataset	Method	Corrected (%)	Miscorrected (%)	Unchanged (%)
CROHME 2016	11-grams	4.01	1.31	94.68
	GRULM_2L	2.96	<b>1.05</b>	95.99
	<b>TMLM_8L</b>	<b>4.62</b>	1.66	93.72
CROHME 2019	11-grams	1.83	1.67	96.50
	GRULM_2L	1.92	<b>1.42</b>	96.66
	<b>TMLM_8L</b>	<b>2.50</b>	1.67	95.83

CROHME 2016 testing set. Combining the language models only improved around 1 percentage point in the case of TAP and PAL\_v2 while TMLM\_8L improves 2.97 percentage points. Here, we cannot conclude that our method for utilizing a math language model is better than their methods since they utilized different types of HME

recognizers as well as different LaTeX corpora to train their language models. We consider conducting more experiments on the combination method as a remaining work.

Table 5.7 shows the recognition results in more detail about the percentage of corrected cases, miscorrected cases, and unchanged cases when combining SRTC\_SLP with three different language models on the CROHME 2016 and CROHME 2019 testing sets. The percentages of the corrected cases by TMLM\_8L are the highest compared to 11-grams and GRULM\_2L on both testing sets. However, that of the miscorrected cases by TMLM\_8L is the worst compared to others. GRULM\_2L caused the least miscorrections compared to others, but it could not correct many cases. 11-grams and TMLM\_8L have comparable percentages of miscorrected cases, but TMLM\_8L corrected more cases than 11-grams did.

### 5.3.2.5. Error analysis

In this section, we present some samples which are corrected or miscorrected when applying TMLM\_8L with the SRTC\_SLP recognizer.

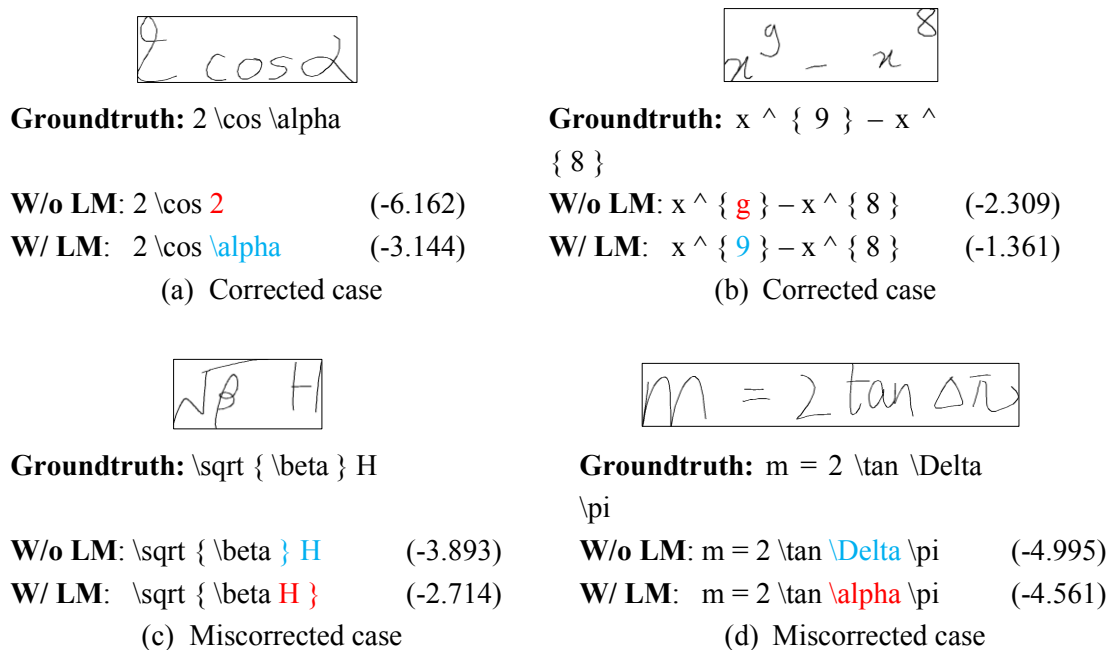


Figure 5.8. Examples of corrected and miscorrected cases when combining the SRTC\_SLP recognizer and TMLM\_8L (LM: language model). Each case shows an HME image, its ground truth, and its recognition candidates with/without TMLM\_8L and their corresponding scores from TMLM\_8L.

Figure 5.8(a) and Figure 5.8(b) show two corrected cases. In Figure 5.8(a), “ $\alpha$ ” in the HME sample are recognized as “2” without using TMLM\_8L since it seems to be similarly written as “2”. However, the language score of the candidate with “ $\alpha$ ” is significantly higher than the one with “2”. Therefore, the recognizer combined TMLM\_8L results in the correct prediction. TMLM\_8L performs well since “ $\alpha$ ” seems

more likely to appear next to the trigonometry function (e.g., sine, cosine, and tangent) than a number. Similarly, “9” in the HME sample of Figure 5.8(b) are correctly recognized by TMLM\_8L.

Figure 5.8(c) and Figure 5.8(d) show two miscorrected cases. The case in Figure 5.8(c) is miscorrected since the language model score of the incorrect result is higher than that of the correct result. We can realize that the context, in this case, is not clear. The case in Figure 5.8(d) is miscorrected since “ $\alpha$ ” seems more likely to appear next to the tangent symbol than “ $\Delta$ ”.

According to those examples, we can see that modeling MEs is still challenging since the context in an ME is not clear and our corpus of MEs might not be enough to estimate the distribution of MEs.

## 5.4. Conclusions

We introduced the Bidirectional Context for symbol classification to solve the problem of recognizing ambiguous symbols. The proposed method improved the symbol recognition rate as compared to the previous approach without context. The proposed method is then integrated with the Stochastic Context-Free Grammar recognition system for the problem of HMEs recognition. We measured the effectiveness of the improvement on the CROHME 2016 dataset and recorded competitive results, from the tasks of symbol segmentation, symbol classification to the task of HMEs recognition. The results showed that the proposed method encode better context to distinguish symbols.

In addition, we presented a transformer-based math language model (TMLM) for improving the recognition rate of HME recognition systems. We showed that our TMLMs perform better than the traditional language models for MEs, i.e., the  $N$ -gram and GRULM. The best perplexity achieved is 4.42, resulted from TMLM\_8L of 8 transformer layers. Combining TMLM\_8L with the online HME recognizer in [89] improved the expression rate by 2.97 and 0.83 percentage points on the CROHME 2016 and CROHME 2019 testing set, respectively.

## **CHAPTER 6. Conclusion and future works**

### **6.1. Conclusions**

In this thesis, we presented our strategy and two tools for collecting and annotating handwritten descriptive answers. They should be important resources for the study on automatic and semi-automatic marking for descriptive questions as well as the study on handwriting recognition methods for the most natural patterns. We currently published the e-testing tool for the research community.

To provide clustering-based marking for handwritten mathematics answers, we proposed two approaches for clustering online handwritten mathematical expressions (OHMEs). We achieved the best results of around 0.916 and 0.915 for purity and around 0.556 and 0.702 for the marking cost on the two answer datasets, Dset\_Mix and NIER\_CBT, respectively. Those values of the marking cost indicate that we can reduce the cost more than 0.298 point than manually marking OHME answers. We also showed that setting the number of clusters according to the number of answers could be beneficial.

Improving OHME recognition can be beneficial for the clustering-based marking. We presented a deep BLSTM\_CTC model with the bidirectional context for symbol classification to solve the problem of recognizing ambiguous symbols. The experimental results showed that the proposed method encode better context to distinguish symbols.

In addition, we presented a transformer-based math language model (TMLM) to solve the problem of recognizing ambiguous symbols. The experimental results showed that our TMLM outperformed the traditional math language models. TMLM can be combined with both online and offline HME recognizers to improve the performance.

### **6.2. Future works**

For the plan of collecting HMAs, we are now preparing the first versions of the e-testing and e-marking tools. Then, we will collect HMAs from some students and provide collected HMAs for collaborators on a web-based hosting service. By releasing the tools publicly and freely, we intend to scale our initial effort to a larger number and variety of participants and volunteers.

For the clustering of OHMEs, we need to evaluate our methods in a large dataset of handwritten answers. In addition, we should consider mini-batch clustering for larger answer sets. User interface for markers is also another problem to study.

For the proposed deep BLSTM-CTC model with the bidirectional context, our model is trained using pre-extracted features, which currently include basic features for each point

of a stroke. We can utilize other features so that the model can represent a better context to recognize the mathematical symbols.

For the math language model, we should enrich the source of ME LaTeX by collecting open sources on the internet. Secondly, we should modify our TMLM to exploit the bidirectional context in MEs. Thirdly, the method for jointly training an HME recognizer and a math language model should be studied for better optimization.

## REFERENCES

- [1] M. Mahdavi, R. Zanibbi, H. Mouchere, C. Viard-Gaudin, U. Garain, CROHME + TFD: Competition on recognition of handwritten mathematical expressions and typeset formula detection, in: Proc. Inter. Conf. Doc. Anal. Recognit., 2019: pp. 1533–1538.
- [2] J.J. LaViola, R.C. Zeleznik, MathPad2: A system for the creation and exploration of mathematical sketches, ACM Trans. Graph. 23 (2004) 432–440.
- [3] K. F. Chan, D. Y. Yeung, PenCalc: a novel application of on-line mathematical expression recognition technology, in: Proc. Int. Conf. Doc. Anal. Recognit., 2001: pp. 774–778.
- [4] T. O’Connell, C. Li, T. S Miller., R. C. Zeleznik, J. J. LaViola, A usability evaluation of AlgoSketch: a pen-based application for mathematics, in: Proc. Eurographics Symp. Sketch-Based Interfaces Model., New York, New York, USA, 2009: pp. 149–157.
- [5] V. T. M. Khuong, H. Q. Ung, C. T. Nguyen, M. Nakagawa, Clustering Offline Handwritten Mathematical Answers for Computer-Assisted Marking, in: Proc. of Inter. Conf. Pattern Recognit. Artif. Intell., Montreal, 2018: pp. 122–126.
- [6] H. Mouchère, R. Zanibbi, U. Garain, C. Viard-Gaudin, Advancing the state of the art for handwritten math recognition: the CROHME competitions, 2011–2014, Int. J. Doc. Anal. Recognit. 2016 192. 19 (2016) 173–189.
- [7] R. Zanibbi, D. Blostein, Recognition and retrieval of mathematical expressions, Int. J. Doc. Anal. Recognit. 15 (2012) 331–357.
- [8] G. Labahn, E. Lank, M. Marzouk, A. Bunt, S. MacLean, D. Tausky, MathBrush: a case study for pen-based interactive mathematics, Proc. Fifth Eurographics Conf. Sketch-Based Interfaces Model. (2008) 143–150.
- [9] A.-M. Awal, H. Mouchère, C. Viard-Gaudin, Towards Handwritten Mathematical Expression Recognition, in: 2009 10th Int. Conf. Doc. Anal. Recognit., 2009: pp. 1046–1050.
- [10] S. Quiniou, H. Mouchère, S.P. Saldarriaga, C. Viard-Gaudin, E. Morin, S. Petitrenaud, S. Medjkoune, HAMEX - a Handwritten and Audio Dataset of Mathematical Expressions, in: 11th Int. Conf. Doc. Anal. Recognit., 2011: pp. 452–456.
- [11] J. Stria, M. Bresler, D. Průša, V. Hlaváč, MfrDB: Database of annotated on-line mathematical formulae, in: Proc. - Int. Work. Front. Handwrit. Recognition, IWFHR, 2012: pp. 542–547.

- [12] F.D.J. Aguilar, N.S.T. Hirata, ExpressMatch: A system for creating ground-truthed datasets of online mathematical expressions, in: 10th IAPR Int. Work. Doc. Anal. Syst. DAS 2012, 2012: pp. 155–159.
- [13] V. T. M. Khuong, K. M. Phan, H. Q. Ung, C. T. Nguyen, M. Nakagawa, Clustering of Handwritten Mathematical Expressions for Computer-Assisted Marking, IEICE Trans. Inf. Syst. E104.D (2021) 275–284.
- [14] C. T. Nguyen, V. T. M. Khuong, H. T. Nguyen, M. Nakagawa, CNN based spatial classification features for clustering offline handwritten mathematical expressions, Pattern Recognit. Lett. 131 (2020) 113–120.
- [15] Z. S. Harris, Distributional Structure, Word. 10 (1954) 146–162.
- [16] Q. Le, T. Mikolov, Distributed Representations of Sentences and Documents, in: Proc. Int. Conf. Mach. Learn., 2014: pp. 1188–1196.
- [17] D. Ienco, R. Interdonato, Deep Multivariate Time Series Embedding Clustering via Attentive-Gated Autoencoder, in: Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), 2020: pp. 318–329.
- [18] Q. Ma, J. Zheng, S. Li, G.W. Cottrell, Learning Representations for Time Series Clustering, in: Adv. Neural Inf. Process. Syst., 2019: pp. 3776–3786.
- [19] S. J. Rao, Y. Wang, G. Cottrell, A Deep Siamese Neural Network Learns the Human-Perceived Similarity Structure of Facial Expressions Without Explicit Categories, in: Proc. 38th Annu. Conf. Cogn. Sci. Soc., 2016: pp. 217–222.
- [20] R. Cummins, M. Zhang, T. Briscoe, Constrained Multi-Task Learning for Automated Essay Scoring, in: Proc. Annu. Meet. Assoc. Comput. Linguist., Stroudsburg, PA, USA, 2016: pp. 789–799.
- [21] V. Salvatore, N. Francesca, C. Alessandro, An Overview of Current Research on Automated Essay Grading, J. Inf. Technol. Educ. Res. 2 (2003) 319–330.
- [22] T. Ishioka, M. Kameda, Automated Japanese essay scoring system:jess, in: Proc. Int. Work. Database Expert Syst. Appl., 2004: pp. 4–8.
- [23] S. Srihari, J. Collins, R. Srihari, H. Srinivasan, S. Shetty, J. Brutt-Griffler, Automatic scoring of short handwritten essays in reading comprehension tests, Artif. Intell. 172 (2008) 300–324.
- [24] S. Basu, C. Jacobs, L. Vanderwende, Powergrading: a Clustering Approach to Amplify Human Effort for Short Answer Grading, Trans. Assoc. Comput. Linguist. 1 (2013) 391–402.
- [25] M. Brooks, S. Basu, C. Jacobs, L. Vanderwende, Divide and correct: Using clusters to grade short answers at scale, in: Proc. ACM Conf. Learn. @ Scale, New York, New York, USA, 2014: pp. 89–98.
- [26] A. Singh, S. Karayev, K. Gutowski, P. Abbeel, Gradescope: A Fast, Flexible, and Fair System for Scalable Assessment of Handwritten Work, in: Proc. ACM Conf. Learn. @ Scale, New York, New York, USA, 2017: pp. 81–88.
- [27] J. Zhang, J. Du, L. Dai, Track, Attend, and Parse (TAP): An End-to-End Framework for Online Handwritten Mathematical Expression Recognition, IEEE Trans. Multimed. 21 (2019) 221–233.
- [28] 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 Recognit. Lett.* 35 (2014) 58–67.
- [29] R. Yamamoto, S. Sako, T. Nishimoto, S. Sagayama, On-Line Recognition of Handwritten Mathematical Expressions Based on Stroke-Based Stochastic Context-Free Grammar, 2006. (accessed February 1, 2021).
- [30] F. Simistira, V. Katsouros, G. Carayannis, Recognition of online handwritten mathematical formulas using probabilistic SVMs and stochastic context free grammars, *Pattern Recognit. Lett.* 53 (2015) 85–92.
- [31] A. D. Le, T. V. Phan, M. Nakagawa, A System for Recognizing Online Handwritten Mathematical Expressions and Improvement of Structure Analysis, in: *Proc. IAPR Int. Work. Doc. Anal. Syst.*, 2014: pp. 51–55.
- [32] A. M. Awal, H. Mouchère, C. V. Gaudin, A global learning approach for an online handwritten mathematical expression recognition system, *Pattern Recognit. Lett.* 35 (2014) 68–77.
- [33] M. Celik, B. Yanikoglu, Probabilistic mathematical formula recognition using a 2D context-free graph grammar, in: *Proc. Int. Conf. Doc. Anal. Recognition, ICDAR*, 2011: pp. 161–166.
- [34] Y. Shi, H. Y. Li, F. K. Soong, A unified framework for symbol segmentation and recognition of handwritten mathematical expressions, in: *Proc. Int. Conf. Doc. Anal. Recognition, ICDAR*, 2007: pp. 854–858.
- [35] M. Koschinski, H.-J. Winkler, M. Lang, Segmentation and recognition of symbols within handwritten mathematical expressions, in: *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, 1995: pp. 2439–2442.
- [36] H.-J. Winkler, M. Lang, On-line symbol segmentation and recognition in handwritten mathematical expressions, in: *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, 1997: pp. 3377–3380.
- [37] L. Hu, R. Zanibbi, MST-based visual parsing of online handwritten mathematical expressions, in: *Proc. Int. Conf. Front. Handwrit. Recognition, ICFHR*, 2016: pp. 337–342.
- [38] A. Kosmala, G. Rigoll, S. Lavirotte, L. Pottier, On-line handwritten formula recognition using hidden Markov models and context dependent graph grammars, in: *Proc. Int. Conf. Doc. Anal. Recognition, ICDAR*, 1999: pp. 107–110.
- [39] F. J. Aguilar, H. Mouchère, C. V. Gaudin, N. S. T. Hirata, Top-down online handwritten mathematical expression parsing with graph grammar, in: *Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, 2015: pp. 444–451.
- [40] Z. Hong, N. You, J. Tan, N. Bi, Residual BiRNN based Seq2Seq model with transition probability matrix for online handwritten mathematical expression recognition, in: *Proc. Int. Conf. Doc. Anal. Recognition, ICDAR*, 2019: pp. 635–640.
- [41] J. Zhang, J. Du, S. Zhang, D. Liu, Y. Hu, J. Hu, S. Wei, L. Dai, Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition, *Pattern Recognit.* 71 (2017) 196–206.
- [42] A. Poznanski, L. Wolf, CNN-N-Gram for Handwriting Word Recognition, in: *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2016: pp. 2305–2314.

- [43] B. Zhu, X.D. Zhou, C.L. Liu, M. Nakagawa, A robust model for on-line handwritten japanese text recognition, *Int. J. Doc. Anal. Recognit.* 13 (2010) 121–131.
- [44] R. Reeve Ingle, Y. Fujii, T. Deselaers, J. Baccash, A.C. Popat, A scalable handwritten text recognition system, in: *Proc. Int. Conf. Doc. Anal. Recognition, ICDAR*, 2019: pp. 17–24.
- [45] J.W. Wu, F. Yin, Y.M. Zhang, X.Y. Zhang, C.L. Liu, Handwritten Mathematical Expression Recognition via Paired Adversarial Learning, *Int. J. Comput. Vis.* 128 (2020) 2386–2401.
- [46] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, Recurrent neural network based language model, in: *11th Annu. Conf. Int. Speech Commun. Assoc.*, 2010: pp. 1045–1048.
- [47] A. Radford, K. Narasimhan, T. Salimans, L. Sutskever, Improving Language Understanding by Generative Pre-Training, [https://S3-Us-West-2.Amazonaws.Com/Openai-Assets/Research-Covers/Language-Unsupervised/Language\\_understanding\\_paper.Pdf](https://S3-Us-West-2.Amazonaws.Com/Openai-Assets/Research-Covers/Language-Unsupervised/Language_understanding_paper.Pdf). (2018). (accessed May 12, 2021).
- [48] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language Models are Unsupervised Multitask Learners, [https://D4mucfpksywv.Cloudfront.Net/Better-Language-Models/Language\\_models\\_are\\_unsupervised\\_multitask\\_learners.Pdf](https://D4mucfpksywv.Cloudfront.Net/Better-Language-Models/Language_models_are_unsupervised_multitask_learners.Pdf). (2019).
- [49] A. Vaswani, G. Brain, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention Is All You Need, in: *Adv. Neural Inf. Process. Syst.*, 2017.
- [50] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, R. Salakhutdinov, Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context, in: *ACL 2019 - 57th Annu. Meet. Assoc. Comput. Linguist.*, 2019: pp. 2978–2988. (accessed May 12, 2021).
- [51] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, Q. V Le, XLNet: Generalized Autoregressive Pretraining for Language Understanding, in: *Adv. Neural Inf. Process. Syst.*, 2019.
- [52] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding - *ACL Anthology*, in: *Proc. 2019 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol.*, 2019: pp. 4171–4186.
- [53] M. Nakagawa, K. Matsumoto, Collection of on-line handwritten Japanese character pattern databases and their analyses, *Int. J. Doc. Anal. Recognit.* 7 (2004) 69–81.
- [54] H.T. Nguyen, C.T. Nguyen, P.T. Bao, M. Nakagawa, A database of unconstrained Vietnamese online handwriting and recognition experiments by recurrent neural networks, *Pattern Recognit.* 78 (2018) 291–306.
- [55] T. Matsushita, M. Nakagawa, A Database of On-Line Handwritten Mixed Objects Named “Kondate,” in: *Int. Conf. Front. Handwrit. Recognition, ICFHR*, 2014: pp. 369–374.
- [56] Stephen M. Watt, Tom Underhill, *Ink Markup Language (InkML)*, 2009.



- [57] H. Q. Ung, V. T. M. Khuong, A. D. Le, C. T. Nguyen, M. Nakagawa, Bag-of-features for clustering online handwritten mathematical expressions, in: Proc. of Inter. Conf. Pattern Recognit. Artif. Intell., 2018: pp. 127–132.
- [58] S. Mori, C.Y. Suen, K. Yamamoto, Historical Review of OCR Research and Development, Proc. IEEE. 80 (1992) 1029–1058.
- [59] S. Mori, K. Yamamoto, M. Yasuda, Research on Machine Recognition of Handprinted Characters, IEEE Trans. Pattern Anal. Mach. Intell. PAMI-6 (1984) 386–405.
- [60] R. Zhao, K. Mao, Fuzzy Bag-of-Words Model for Document Representation, IEEE Trans. Fuzzy Syst. 26 (2018) 794–804.
- [61] M. D. El-Din, Enhancement Bag-of-Words Model for Solving the Challenges of Sentiment Analysis, Int. J. Adv. Comput. Sci. Appl. 7 (2016) 244–252.
- [62] B. Aljaber, N. Stokes, J. Bailey, J. Pei, Document clustering of scientific texts using citation contexts, Inf. Retr. Boston. 13 (2010) 101–131.
- [63] D. François, V. Wertz, M. Verieysen, The concentration of fractional distances, IEEE Trans. Knowl. Data Eng. 19 (2007) 873–886.
- [64] S. I. Nikolenko, [1909.11512] Synthetic Data for Deep Learning, (n.d.). (accessed March 6, 2020).
- [65] K. M. Phan, V. T. M. Khuong, H. Q. Ung, M. Nakagawa, Generating Synthetic Handwritten Mathematical Expressions from a LaTeX Sequence or a MathML Script, in: Proc. Int. Conf. Doc. Anal. Recognit., 2020: pp. 922–927.
- [66] C. T. Nguyen, T. N. Truong, H. Q. Ung, M. Nakagawa, Online Handwritten Mathematical Symbol Segmentation and Recognition with Bidirectional Context, in: Proc. Int. Conf. Front. Handwrit. Recognit., 2020: pp. 355–360.
- [67] H. Mouchere, C. Viard-Gaudin, R. Zanibbi, U. Garain, ICFHR2016 CROHME: Competition on Recognition of Online Handwritten Mathematical Expressions, in: 2016 15th Int. Conf. Front. Handwrit. Recognit., 2016: pp. 607–612.
- [68] M. Mahdavi, R. Zanibbi, Tree-Based Structure Recognition Evaluation for Math Expressions: Techniques and Case Study, in: Proc. IAPR Int. Work. Graph. Recognit., Sydney, 2019.
- [69] D. Arthur, S. Vassilvitskii, k-means++: the advantages of careful seeding, in: Proc. Annu. ACM-SIAM Symp. Discret. Algorithms, 2007: pp. 1027–1035.
- [70] P. J. Rousseeuw, Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, J. Comput. Appl. Math. 20 (1987) 53–65.
- [71] J. A. Hartigan, Clustering Algorithms, 1975.
- [72] J. Chang, L. Wang, G. Meng, S. Xiang, C. Pan, Deep Adaptive Image Clustering, in: Proc. IEEE Int. Conf. Comput. Vis., 2017: pp. 5880–5888.
- [73] H.Q. Ung, C.T. Nguyen, K.M. Phan, V.T.M. Khuong, M. Nakagawa, Clustering online handwritten mathematical expressions, Pattern Recognit. Lett. 146 (2021) 267–275.
- [74] F. Yasuno, K. Nishimura, S. Negami, Y. Namikawa, Development of Mathematics Items with Dynamic Objects for Computer-Based Testing Using Tablet PC, Int. J. Technol. Math. Educ. 26 (2019) 131–137.

- [75] X. Y. Zhang, F. Yin, Y. M. Zhang, C. L. Liu, Y. Bengio, Drawing and Recognizing Chinese Characters with Recurrent Neural Network, *IEEE Trans. Pattern Anal. Mach. Intell.* 40 (2018) 849–862.
- [76] K. Cho, B. V. Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, in: *Conf. Empir. Methods Nat. Lang. Process.* 2014, 2014: pp. 1724–1734.
- [77] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, J. Schmidhuber, A Novel Connectionist System for Unconstrained Handwriting Recognition, *IEEE Trans. Pattern Anal. Mach. Intell.* 31 (2009) 855–868.
- [78] A. Graves, J. Schmidhuber, Framewise phoneme classification with bidirectional LSTM and other neural network architectures., *Neural Networks.* 18 (2005) 602–10.
- [79] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, *Neural Comput.* 9 (1997) 1735–1780.
- [80] M. Oquab, L. Bottou, I. Laptev, J. Sivic, Is object localization for free? - Weakly-supervised learning with convolutional neural networks, in: *2015 IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015: pp. 685–694.
- [81] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, A. Torralba, Learning Deep Features for Discriminative Localization, in: *2016 IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016: pp. 2921–2929.
- [82] S. Jaeger, S. Manke, J. Reichert, A. Waibel, Online handwriting recognition: The NPen++ recognizer, *Int. J. Doc. Anal. Recognit.* 3 (2001) 169–180.
- [83] F. Alvaro, J.A. Sanchez, J.M. Benedi, Offline features for classifying handwritten math symbols with recurrent neural networks, in: *Proc. - Int. Conf. Pattern Recognit.*, 2014: pp. 2944–2949.
- [84] U. Ramer, An iterative procedure for the polygonal approximation of plane curves, *Comput. Graph. Image Process.* 1 (1972) 244–256.
- [85] R. Al-Rfou, D. Choe, N. Constant, M. Guo, L. Jones, Character-level language modeling with deeper self-attention, in: *33rd AAAI Conf. Artif. Intell.*, 2019: pp. 3159–3166.
- [86] J.L. Ba, J.R. Kiros, G.E. Hinton, Layer Normalization, [Http://Arxiv.Org/Abs/1607.06450](http://arxiv.org/abs/1607.06450). (2016). (accessed May 12, 2021).
- [87] A.D. Le, M. Nakagawa, A system for recognizing online handwritten mathematical expressions by using improved structural analysis, *Int. J. Doc. Anal. Recognit.* 19 (2016) 305–319.
- [88] F. Julca-Aguilar, N.S.T. Hirata, C. Viard-Gaudin, H. Mouchere, S. Medjkoune, Mathematical Symbol Hypothesis Recognition with Rejection Option, in: *14th Int. Conf. Front. Handwrit. Recognit.*, 2014: pp. 500–505.
- [89] C.T. Nguyen, N.-T. Truong, H.T. Nguyen, M. Nakagawa, Global Context for improving recognition of Online Handwritten Mathematical Expressions, in: *Proc. Int. Conf. Doc. Anal. Recognition, ICDAR*, 2021.
- [90] J. Cocke, J. T. Schwartz, *Programming Languages and Their Compilers: Preliminary Notes*, 1970. (accessed May 13, 2021).

- [91] E. Grave, A. Joulin, M. Cisse, D. Grangier, H. Jegou, Efficient softmax approximation for GPUsÉdouard, in: 34th Int. Conf. Mach. Learn., 2017: pp. 1302–1310.
- [92] I. Loshchilov, F. Hutter, Decoupled Weight Decay Regularization, in: Int. Conf. Learn. Represent., 2019.
- [93] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, A.M. Rush, HuggingFace’s Transformers: State-of-the-art Natural Language Processing, [Http://Arxiv.Org/Abs/1910.03771](http://Arxiv.Org/Abs/1910.03771). (2019). (accessed May 13, 2021).
- [94] H. Dai Nguyen, A.D. Le, M. Nakagawa, Deep neural networks for recognizing online handwritten mathematical symbols, in: Proc. - 3rd IAPR Asian Conf. Pattern Recognition, ACPR 2015, 2016: pp. 121–125.
- [95] K. Davila, S. Ludi, R. Zanibbi, Using Off-Line Features and Synthetic Data for On-Line Handwritten Math Symbol Recognition, in: Proc. Int. Conf. Front. Handwrit. Recognition, ICFHR, 2014: pp. 323–328.

## APENDIX I – PUBLICATIONS

### Journal papers

1. **H. Q. Ung**, C. T. Nguyen, K. M. Phan, V. T. M. Khuong, and M. Nakagawa, “**Clustering online handwritten mathematical expressions**”, Pattern Recognition Letters, Vol. 146, pp. 267-275, 2021.
2. V. T. M. Khuong, K. M. Phan, **H. Q. Ung**, C. T. Nguyen, and M. Nakagawa, “**Clustering of Handwritten Mathematical Expressions for Computer-assisted marking**”, IEICE Transactions on Information and Systems, Vol.E104-D, pp. 275-284, 2021.
3. H. Duong, **U. Q. Huy**, P. T. Bao, J. Y. Kim, "A Method for Selecting the Most Informative Iris Image from Real Time Video Stream", The Journal of Korean Institute of Information Technology, Vol. 12, No. 9, pp. 61-67, 2014.

### International Conference Papers

1. **H. Q. Ung**, C. T. Nguyen, H. T. Nguyen, T.-N. Truong, and M. Nakagawa, “**A Transformer-based Math Language Model for Handwritten Math Expression Recognition**”, in Proceedings of ICDAR Workshop on Document Images and Language, pp. 403-415, Lausanne, 2021.
2. **H. Q. Ung**, C. T. Nguyen, H. T. Nguyen, and M. Nakagawa, “**GSSF: A Generative Sequence Similarity Function based on a Seq2Seq Model for Clustering Online**

- Handwritten Mathematical Answers**”, in Proceedings of International Conference on Document Analysis and Recognition, pp. 145-159, Lausanne, 2021.
3. **H. Q. Ung**, K. M. Phan, M. Nakagawa, “**Strategy and Tools for Collecting and Annotating Handwritten Descriptive Answers for Developing Automatic and Semi-Automatic Marking - an Initial Effort to Math**”, in Proceedings of 2nd International Workshop on Open Services and Tools for Document Analysis, pp. 13-18, Sydney, 2019.
  4. **H. Q. Ung**, V. T. M. Khuong, A. D. Le, C. T. Nguyen, M. Nakagawa, "**Bag-of-Features for Clustering Online Handwritten Mathematical Expressions**," in Proceedings of International Conference on Pattern Recognition and Artificial Intelligence, pp. 127-132, Montreal, 2018.

### Joint Works

1. T.-N. Truong, **H. Q. Ung**, H. T. Nguyen, C. T. Nguyen, and M. Nakagawa, “**Relation-Based Representation for Handwritten Mathematical Expression Recognition**”, in Proceedings of 14th IAPR International Workshop on Graphics Recognition, pp. 7-19, Lausanne, 2021.
2. C.T. Nguyen, T.-N. Truong, **H. Q. Ung**, and M. Nakagawa “**Online Handwritten Mathematical Symbol Segmentation and Recognition with Bidirectional Context**”, in Proceedings of 17th International Conference on Frontiers in Handwriting Recognition, pp. 355-360, Dortmund, 2020.
3. K. M. Phan, V. T. M. Khuong, **H. Q. Ung**, M. Nakagawa, “**Generating Synthetic Handwritten Mathematical Expressions from a LaTeX Sequence or a MathML Script**”, in Proceedings of International Conference on Document Analysis and Recognition, pp. 922-927, Sydney, 2019.
4. V. T. M. Khuong, **H. Q. Ung**, C. T. Nguyen, M. Nakagawa, "**Clustering Offline Handwritten Mathematical Answers for Computer-Assisted Marking**", in Proceedings of International Conference on Pattern Recognition and Artificial Intelligence, pp. 127-132, Montreal, 2018.
5. D. M. H. Nguyen, H. T. Vu, **H. Q. Ung**, B. T. Nguyen, "**3D-brain segmentation using deep neural network and Gaussian mixture model**", in Proceedings of IEEE Winter Conference on Applications of Computer Vision, pp. 815-824, Santa Rosa, 2017.

### Patents

中川正樹, ヴ・トラン・ミン・クオン, ウン・コアン・フィ. プログラム, 情報記憶媒体及びクラスタリング装置, 特願 2018-082354.