Pattern Recognition Letters 35 (2014) 46–57

Contents lists available at SciVerse ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

An annotation assistance system using an unsupervised codebook composed of handwritten graphical multi-stroke symbols

Jinpeng Li *, Harold Mouchère, Christian Viard-Gaudin

IRCCyN (UMR CNRS 6597), L'UNAM, Université de Nantes, France

ARTICLE INFO

Article history:
Available online 13 December 2012

Keywords:
Graphical symbol knowledge extraction
Graphical symbol retrieval
Spatial relations
Minimum Description Length principle
On-line handwriting

ABSTRACT

Many present recognition systems take advantage of ground-truthed datasets for training, evaluating and testing. But the creation of ground-truthed datasets is a tedious task. This paper proposes an iterative unsupervised handwritten graphical symbols learning framework which can be used for assisting such a labeling task. Initializing each stroke as a segment, we construct a relational graph between the segments where the nodes are the segments and the edges are the spatial relations between them. To extract the relevant patterns, a quantization of segments and spatial relations is implemented. Discovering graphical symbols becomes then the problem of finding the sub-graphs according to the Minimum Description Length (MDL) principle. The discovered graphical symbols will become the new segments for the next iteration. In each iteration, the quantization of segments yields the codebook in which the user can label graphical symbols. This original method has been first applied on a dataset of simple mathematical expressions. The results reported in this work show that only 58.2% of the strokes have to be manually labeled.

1. Introduction

Graphical symbols which are the lexical units of graphical languages are composed of a spatial layout of single or several strokes. Usually everybody share some conventions about the symbol shape. These conventions allow individuals to read graphical messages comprising similar symbols. Many existing recognition systems (Tappert et al., 1990) analogously require the definition of the character or symbol set, and rely on a training dataset which defines the ground-truth at the symbol level. A machine learning algorithm in recognition systems consequently can be trained to recognize symbols from large, realistic corpora of ground-truthed input. Such datasets are essential for the training, evaluation, and testing stages of the recognition systems. However, collecting all the ink samples and labeling them at the symbol level is a very long and tedious task. Hence, it would be very interesting to be able to assist this process, so that most of the tedious work can be done automatically, and that only a high level supervision need to be defined to conclude the labeling process.

In this regard, we propose to extract automatically a finite set of relevant patterns, called codebook within an unlabeled dataset. Searching relevant patterns and extracting them aim to reduce the redundancy in appearance of basic regular shapes and regular layout of these shapes in a large collection of handwritten scripts.

For the targeted application, which is related to an on-line handwritten corpus of mathematical numerical expressions, we consider that the basic units are the strokes, a sequence of points between a pen-down and a pen-up. Should this assumption not be verified, then an additional segmentation process will have to be undergone, so that every basic graphical unit belongs to a unique symbol. Conversely, a symbol can be made of one or several strokes, which are not necessarily drawn consecutively, i.e. we do not exclude interspersed symbols. Afterward, a symbol is made of a single stroke or several strokes within the confines of specific spatial composition. The problem is to identify symbols from a large collection of handwritten strokes in spatial layouts. Let us illustrate some simple examples to understand the problems.

Imagine a document with only two different shapes of stroke, e.g. "−" and ">". Without any context, "−" and ">" might be regarded as two different symbols "minus" and "greater than" respectively. Each stroke corresponds directly to a single symbol. If two strokes are placed together like "→" we can imagine it becomes another symbol "arrow". A stroke is only a part of symbol. Eventually, the same kind of stroke according to the context will be either a single symbol or a piece of a more complex symbol. So the first problem pointed out is searching different shapes of strokes, termed as *graphemes*.

Let us put two strokes together: it exists many composition rules named *spatial relations*. Applying two same graphemes, two

* Corresponding author.
E-mail addresses: jinpeng.li@univ-nantes.fr (J. Li), harold.mouchere@univ-nantes.fr (H. Mouchère), christian.viard-gaudin@univ-nantes.fr (C. Viard-Gaudin).

0167-8655/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.patrec.2012.11.018

different symbols, "≻" and "→", can be constructed. The only difference between them is that "−" is arranged on the right side in "≻" while on the left side in "→". This left and right relation is easily defined manually.

It is possible to design new symbols made of more different graphemes and spatial relations. For instance, a new symbol "↔" is constructed using the grapheme set $\{<, -, >\}$. We can say that "−" is *between* "<" and ">". In this case, *between* implies a relationship among three strokes which is the cardinality of this spatial relation (Clementini, 2009). In this paper the cardinality of spatial relation is limited to two strokes: from a reference stroke to an argument stroke; that is a pairwise spatial relation. However, with only 3 strokes we have to consider 6 different pairs of strokes to envisage all appropriate alternatives, for example ("<","−"), ("−", "<"), ("<",">"), etc. The number of spatial relation couples will grow rapidly with the increasing number of strokes in a layout (Li et al., 2011). Searching automatically different pairwise *spatial relations* will be the second problem.

Considering a more complicated example, Fig. 1(a) shows four different symbols, "arrow", "connection", "process", and "terminator". However, the ground-truths are unknown in advance. To avoid the ambiguity that some strokes share the same grapheme, the stroke is referenced by their index $(\cdot)$. Which set of strokes (a segment) represents a symbol? Why the combination of the strokes {(1),(2),(3)} is a valid symbol (actually "arrow")? An intuitive answer is that the spatial composition is "frequent"; it exists two similar patterns in the layout, {(1),(2),(3)} and {(5),(6),(7)}, comprising same graphemes and same spatial relations respectively (which are from the previous two problems). But the equally frequent combination of less strokes {(1),(2)} does not mean a symbol. Moreover, the third arrow {(11),(12)} only contains two strokes but its shape is similar with the previous two arrows. Graphical symbols with the same ground-truth can contain different number of strokes and different graphemes. Hence, the third problem is how to search some repetitive patterns in a layout yielding to the *graphical symbols*. A segmentation will therefore be generated at the symbol level.

By grouping graphemes in segments, we obtain a small finite set of symbol hypothesis called codebook with a higher semantic level. This codebook requires less annotation operations like in Fig. 1(b): only 3 segments have to be labeled instead of 6 symbols including 13 strokes in Fig. 1(a). But all similar segments in a cluster of the codebook do not contain the same ground-truth: different symbols can be mixed in one cluster. For instance, the stroke (4) of symbol "connection" and the stroke (13) of symbol "terminator" are merged in the same cluster because of two similar shapes. The ground-truth not only depends on the similar shape but also depends on context and meaning. Annotating the segments in a codebook will be the fourth problem.

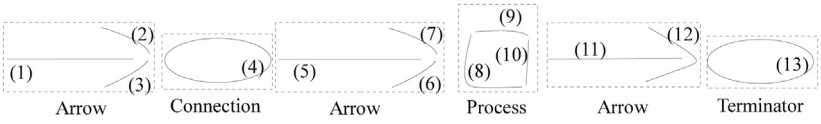
Our previous work Li et al. (2011) studies the unsupervised symbol segmentation using the MDL principle and Li et al. (2012) is specified for the spatial relation learning. This paper proposes to use the unsupervised symbol segmentation using the MDL principle to reduce symbol labeling cost. Section 2 gives a brief survey of cluster labeling in text and in off-line characters, of codebook generation using the unsupervised natural language learning built on two-dimensional spatial relations, and of the annotation on a codebook. The proposed learning framework is revealed in Section 3. In this framework, we extract the codebook composed of multi-stroke symbols which the user can label. Section 5 describes an annotation measure to evaluate the performance on the on-line handwriting corpora. At the end, the conclusion of this work is presented in Section 6.

2. State of the art

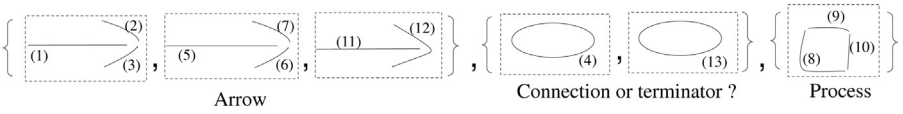
According to authors knowledge there is no existing work about unsupervised symbol extraction on on-line handwriting for annotation assistance. However, several related works will be discussed in this section: reducing annotation workload, handwriting grapheme extraction, and graphical symbol analysis.

2.1. Reducing annotation workload

Unsupervised annotation system already exists on text corpora; it partitions a large collection of text segments into clusters, and then labels each cluster automatically. Many work focus on extracting the label candidates or some keywords from the collection of text segments (Treeratpituk and Callan, 2006). However extracting the label candidates on handwritten graphical corpora in text format would be too difficult without recognition systems; our goal is to annotate the symbols from a raw handwritten dataset so that the recognition systems can be trained on them. Our work focus therefore on grouping the handwritten scripts into several clusters, and then labeling them manually. A similar offline handwriting annotation system Vajda et al. (2011) proposes the idea to label a large number of isolated characters; clustering them into several clusters of characters, and labeling the clusters in order to reduce the human effort. This work shows that over 80% symbol



(a) A handwritten flowchart contains four different graphical symbols, "arrow", "connection", "process", and "terminator" but these symbols are unknown. Searching the undiscovered symbols and grouping them into a small set can ease greatly the annotation workload. Note that all the strokes are marked by the index (.) to avoid the ambiguity



(b) A generated code book from the handwritten flowchart in Figure 1a includes three sets of segments. Some sets of segments represent only one symbol while different symbols are mixed in the others.

Fig. 1. Reducing the human effort on labeling symbols.