On-Line Handwritten Documents Segmentation

J. Blanchard, T. Artières
LIP6, Université Paris 6,
8 rue du capitaine Scott
75015, France
Julien.Blanchard@poleia.lip6.fr, Thierry.Artières@lip6.fr

Abstract

This work concerns note-taking applications; it deals with poorly structured on-line handwritten documents segmentation such as pages of handwritten notes. We extend an existing system based on Probabilistic Feature Grammars. The probabilistic nature of this system allows considering lots of segmentation hypothesis, which is an advantage for poorly structured documents processing, but it goes with important algorithmic complexity. Our improvements concern the handling of this complexity using genetic algorithms, the definition of performance measurements that are adapted to the segmentation of on-line documents, and the evaluation of this segmentation approach on a collection of documents of various qualities.

Keywords: Segmentation, Poorly structured documents, Probabilistic grammars, Genetic algorithms

1. Introduction

This work deals with electronic note-taking applications using a pen based interface. Some of these applications use the pen as an input device only and input signals are immediately recognized and interpreted. Some other applications consider the electronic ink as a new kind of format and allow producing and editing on-line handwritten documents. In this context, a number of tools are needed to manipulate and process such on-line handwritten documents, segmentation tools are among the first ones.

A number of techniques have been developed for off-line documents segmentation such as newspapers or table of contents [4], [6], and [7]. However, since these methods have been designed for a particular class of rather uniform documents, these rely on global features computed on the whole document and are particularly well adapted to printed documents with strong regularities (e.g. uniform line slope or the inter-line size).

A few segmentation methods have been proposed in the last few years to handle handwritten documents corresponding to note-taking applications (on-line and off-line). First methods were based on histogram projections for detecting lines, then to locate words or word groups [5], [8]. Again, those methods were based on rather strong hypothesis about document regularity, particularly for the line slope or the inter-line size.

More recent works [3], [9] dealing with few kinds of documents try to give-up too strict constraints and use contextual information to achieve a local treatment of signal. The system presented in [1] was our first try to answer the two sides of the problem, by being more flexible about features regularity and more generic about type of documents. Formalism chosen is based on Probabilistic Features Grammars (PFGs) [2], which were adapted to two-dimensional datasets. Principal interest of those grammars is that they permit to simply take into account contextual information, which is very important in our case, because poorly structured documents are often ambiguous.

The paper is organized as follows. We first briefly review the core of the method and describe PFG formalism. Then, we detail our main contributions, in particular the use of beam search and of genetic algorithms to break algorithmic complexity. Next, we present experimental results. To do this we define a few performance criteria that are adapted to the case of on-line documents segmentation task. We compare our approach to a more classical method inspired by the Docstrum method [7] on a handwritten documents corpus that we collected in our lab and that consists of both poorly and highly structured documents.

2. Using PFGs for handwritten documents analysis

In this section, we briefly describe the application of Probabilistic Feature Grammars to on-line documents segmentation, more details concerning §1 and §2 may be found in [1]. Our new contributions to this system concern mainly §2.3 and §2.4. In the following, we first describe a grammar for simple texts then we discuss of the probabilistic laws associated to rule production. Next we discuss the training step and present the segmentation

Proceedings of the 9th Int'l Workshop on Frontiers in Handwriting Recognition (IWFHR-9 2004)
0-7695-2187-8/04 $20.00 © 2004 IEEE

algorithm and the techniques used to break down its algorithmic complexity.

2.1. A grammar for simple texts

The grammar we use here is rather simple but it allows us comparing our approach to other segmentation techniques. However, our system is not limited to such grammars and dealing with a new kind of documents requires a limited work: the definition of the grammar and of rules production probabilities. The grammar used here is defined by the five rules:

1) Page → Page [Above] Paragraph
2) Page → [Up Border Above] Paragraph
3) Paragraph → Paragraph [Above] Line
4) Paragraph → Line
5) Line → Line [On-Right-Of] word
6) Line → [Left Border On-Right-Of] word

According to this grammar, a document is a series of paragraphs. Similarly, a paragraph is a succession of lines, and a line is a succession of words (we call word a handwriting signal between a pen-down and a pen-up moves). Note that there is a simple hierarchy in the terms of the grammars. Words are aggregated to form lines that are aggregated to form paragraphs. We will talk of composite term to express this idea: words are composite terms of lines, which themselves are composite terms of paragraphs.

As one may sees, there are some relative spatial operators in these rules. For example a line may be built from an existing line by adding a word that is *[On the right]* of this line. There are also absolute spatial operators. For example, the second rule that allows building lines is the initialization rule: According to this rule and to the operator *[Left Border on the right]*, a line may begin with a word that has nothing on its left. Spatial operators are used at different levels. First they are used as bottom-up operators that allow limiting the number of rules to be activated. For example, not all initialization lines (consisting of any word of a document) are built, only those corresponding to words that are reasonable candidates (i.e. with almost nothing on its left) are built. Second, these operators condition the nature of the probability law for activating rules. For example, the probability that rule 3 be activated, from an existing paragraph *P* and an existing line *L*, is defined as a function of relative positions of *P* and *L* (*P* being above *L* gives higher probability). Some geometric algorithms, based on convex hulls of entities (e.g. *P* and *L*), are used to define how much an entity is above, below, on the right, on the left, of another entity.

2.2. Probability of rule production

PFGs allow integrating contextual informations through propagation of features associated to derived entities. In PFGs, rules are of the form:

$$A=(a_1,a_2,...,a_g)\rightarrow B=(b_1,b_2,...,b_g),C=(c_1,c_2,...,c_g)$$

Where A, B and C are terms (terminals or nonterminals), $(a_1,a_2,...,a_g)$, $(b_1,b_2,...,b_g)$ and $(c_1,c_2,...,c_g)$ are feature vectors (with *g* features) associated to the terms. In our case terms may be paragraphs, lines and words. Such grammars may be viewed as generative stochastic models.

Let note *C(X)* the feature vector associated to a term *X*. The probability to activate rule $A\rightarrow B,C$, then to produce *B* and *C* from *A*, is given by:

$$P(A\rightarrow B,C)=P(C(B),C(C)/C(A))$$

Such probabilities are computed based on the features of the terms and we considered three kinds of features for a particular term.

- its own features (height, width, slope...), we note *f1(X)*.
- the mean features of composite terms, *f2(X)*.
- features describing spatial relationships between composite terms, *f3(X)*.

Hence, a feature vector for a term *X* is given by: *C(X)=(f1(X), f2(X), f3(X))*. For example, a paragraph has its own features (height, width and slope), but is also characterized by mean features of its lines (mean slope, mean width...) and by features describing spatial relationships between its lines (mean distance between two successive lines).

Probabilistic laws for rule activation are defined on such feature vectors as discussed in [1] where features are assumed independent. For example to add a line to a paragraph with the third rule, distance between this line and the last line of paragraph must be close to the mean inter-line feature of the paragraph, the line slope must be closed to the mean line slope of the paragraph etc.

2.3. Parameters learning

Since we use an independence assumption between features, we only discuss here the learning of parameters for a probabilistic law for one particular feature. All these laws (for each feature) are assumed Gaussian. In [1], parameters of these laws were empirically estimated on a collection of documents. In this study, we tried to improve the system with a statistical learning of these parameters. Learning is based on a collection of labeled documents; a labeled document is a document whose parsing is known. Then, for all rules, we may compute means and variance of term features.

2.4. Segmentation: Optimal derivation tree

Page segmentation produces a derivation tree. Each tree node corresponds to a term *t* (with its features) and a rule number *n*, where term *t* was derived with rule *n* based on children nodes terms. Page segmentation is computed with a dynamic programming algorithm that finds optimal derivation tree, i.e. derivation tree with maximal probability. We used originally an algorithm that was close to the one proposed by [10] but we moved towards a bottom-up algorithm since it is much easier to integrate beam search strategy or optimization heuristics in this kind of algorithms.

Our algorithm processes level by level, beginning by building all possible lines from all the words in the page, then building all possible paragraphs from the lines found at the first step, etc. For each level (e.g. lines), the algorithm builds iteratively terms in the same way that in dynamic programming like Viterbi. For example, to build lines, first all beginning lines are built using rule 5, then using iteratively rule 4, all these lines may be extended until there is no more words to aggregate to existing lines.

Unfortunately, serious algorithmic complexity appears since the number of hypotheses to consider quickly becomes huge with the size of the document, recalling that the segmentation algorithm consists in finding simultaneously the reading order of words and their aggregation in structured terms (lines, paragraphs). The consequence of this two dimensional search is a dramatic combinatorial explosion. Even with bottom-up geometric operators as described in §2.1, such an algorithm may deal with a page of a maximum of 10 to 20 "words". For example, for a document with *i* lines of *n* strokes, each stroke having k neighbors, complexity is $O(i*k^n)$.

To deal with these problems, we had to improve our dynamic programming algorithm. We firstly implemented beam search strategies, and secondly integrated a genetic algorithm in the parsing algorithm.

Beam search (i.e. pruning) strategy is used to give up as soon as possible all hypotheses with low probabilities. However, the problem in our case is that we must prune terms that do not correspond to the same number of words and that do not include all same words, so that their likelihoods are very difficult to compare directly. For example two lines, the first one of 5 words and the second one of 8 words, may share one same word. Ideally we would like to remove one of these hypotheses but the scores for the two lines may be incomparable.

We identified a few cases where beams could be used easier, we detail two cases now. The first case consists in the comparison of terms very similar, for example all new lines built from an existing line and different words through rule 4. A pruning may be done within these terms; terms with lowest probabilities are pruned. We use another beam function when the decoding step of a level is finished, looking at reasonable terms. It is interesting here to take into account the frequency of words in all derived terms. If a particular word appears in only one line l, then not only line l must be kept, but all lines including the other words included in l may be pruned.

Such beam pruning strategies help resolve partially the combinatorial problems but are not efficient enough to deal with realistic documents. Thus, we decided to add another pruning strategy, of a different kind, at the end of each level (line, paragraph, etc.) to prune more deeply the hypotheses. When the processing of a level is finished (e.g. all possible lines have been derived), and before going to the next level, we try to prune again in order to keep sets of terms that are consistent in the next level. A consistent set of lines stands for a set of lines that cover a maximum of words of the document and that do not include the same words (null intersection). The idea of the pruning here is to keep terms belonging to consistent sets only. Thus we use a genetic algorithm to select entities collections that better fit to our partitioning condition, it is a classical suboptimal technique.

The initial population is a collection of coherent terms sets randomly found. Each individual is a coherent set of terms. We use the three classical fundamentals operations, crossing, mutation and selection. To cross two sets of terms, we compute a new consistent set using randomly chosen terms from the two sets. Mutation consists in replacing one of entities by another randomly elected from all possible. Last, the fitness function is the percentage of words found.

In order to investigate the efficiency of such a genetic algorithm, we measured at each level the percentage of right terms that were kept and of wrong terms that were deleted, table 1 reports these results. One can see that about 99% of right lines are kept while 76% of wrong lines are pruned, this means that very few right lines were deleted. Results at the paragraph level are similar (respectively 95% and 84%). The main result of this pruning strategy lies in the size of the documents that our system may handle efficiently now. Thanks to this, we may deal now with normal size documents consisting of hundreds of words, without loose in performance.

Right lines kept	Wrong lines deleted	Right paragraphs kept	Wrong paragraphs deleted
99%	76%	95%	84%

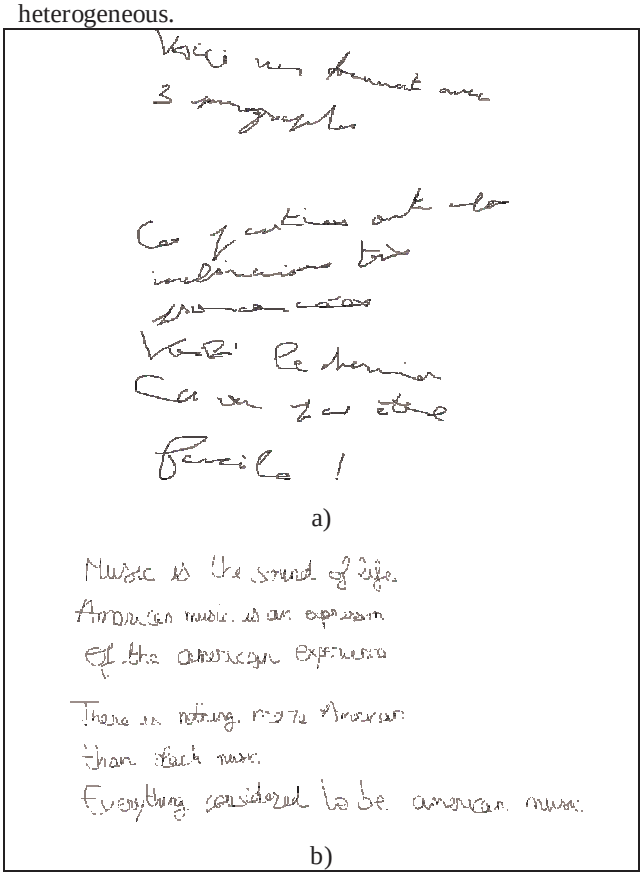
Table 1. Genetic algorithm efficiency.

3. Experimental results

3.1. Database of on-line handwritten documents

We collected, in our laboratory (LIP6), a database of on-line handwritten notes that we labeled manually into lines and paragraphs. Documents were written by a few writers, on a tablet and without any constraints. They consist in up 3 to 30 lines, each line consists in 2 to 20 "words". Documents are divided into two categories (see Figure 1). The first category is a set of "homogeneous" pages, which can be letters, note taking, etc. These homogeneous documents have regular global features, an almost uniform character size, line slope, etc. Documents of the second category are "heterogeneous" pages like drafts or "post-it" with much more varying features.

Experimental results were computed on a database of 56 documents collection, half are homogeneous, half are heterogeneous.



a)

b)

Figure 1. Samples of "heterogeneous" document (a) and "homogeneous" document (b).

3.2. Performance measurement

Since the segmentation result is a parse tree, we might use a tree to tree distance to evaluate the performance of our segmentation procedure. However, such a distance may not be the best performance measure since two trees with very different structures could represent indeed similar segmentations. We rather chose to define performance criteria that are adapted to on-line documents (for which the reading order is important), and that allow evaluating the system behavior at different levels. To deal with on-line documents, we define criteria inspired from the edit-distance that take into account the sequential information. To investigate what is going on at different levels, we defined criteria at the line level and at the paragraph level. We begin with line-based criteria.

To estimate line detection performances we begin to pair off all discovered lines with real (i.e. labeled) lines. Then, we compute two criteria, *L1* and *L2*. *L1* is a between sets distance, it is defined as the percentage of real words that belong to the discovered line. *L2* takes into account the reading order. It is the edit-distance (lines are considered as sequences of words) between the real line and the discovered one. We compute the percentage of elementary operations (numbers of insertions, deletions and substitutions) needed to transform the real line into the discovered line.

At the paragraph level, we pair off the real lines and discovered lines (as found in line level) and then we do similarly to pair discovered and real paragraphs. Then we compute the edit distance between discovered paragraphs and real paragraphs, we note this criterion *P1*. It corresponds to the system ability to aggregate lines into correct paragraphs. Thus, even if lines are not perfectly detected, *P1* may be high, provided lines are well gathered into paragraphs. Finally, at the page level, we use two criteria (*D1* and *D2*). *D1* is computed after lines have been paired off, with the edit distance between the real document and the discovered ones, where these documents are seen as sequences of lines. The second criterion *D2* is an edit distance again, but it compares two documents represented as sequences of paragraphs.

3.3. Reference method

We compare in the following our approach with a reference method. We did not find any method dealing with on-line handwritten documents segmentation that was enough described in the literature. Thus, we chose to implement an off-line method which is flexible enough to be adapted to heterogeneous documents, the Docstrum method [7]. This technique is not based on orientation or line slope and does not require global knowledge about characters size or inter-line space, at the opposite of other classical approaches using histograms or Hough transforms.

Docstrum is based on terms partitioning by the k nearest neighbors algorithm. Segmentation is "bottom-

up", i.e. one considers first low level elements to build words, lines and then blocks. Orientation, policy size, space between lines are estimated from nearest neighbors' angle and distance distributions. Thus, the method we implemented is inspired of Docstrum and was adapted to on-line documents.

3.4. Results

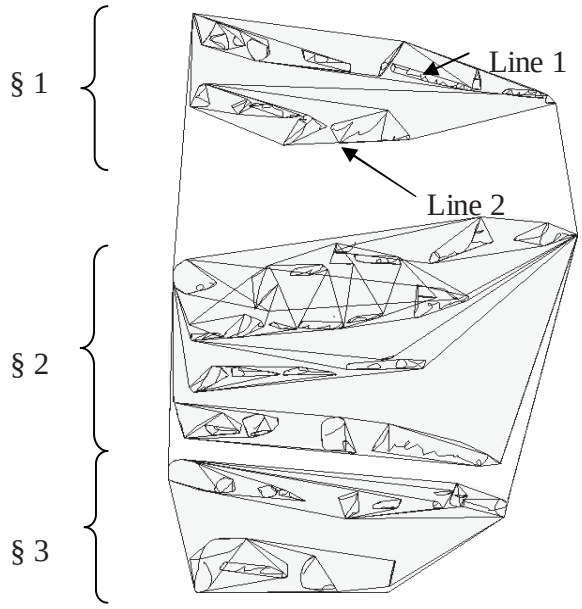
We present in this section experimental results obtained by both our system and the reference method.

%	Complete database		Homogeneous documents		Heterogeneous documents	
	Docst	PFG	Docst	PFG	Docst	PFG
L_1	11.8	**3.8**	*3.4*	*3.9*	22.5	**3.4**
L_2	22.7	**13.1**	6.5	6.9	37.0	**15.7**
P_1	15.3	**2.1**	1.9	2.6	22.8	**5.4**
D_1	29.4	**25.5**	**18.5**	20.7	36.1	**26.2**
D_2	*15.3*	*15.2*	**10.6**	19.4	15.0	**9.0**

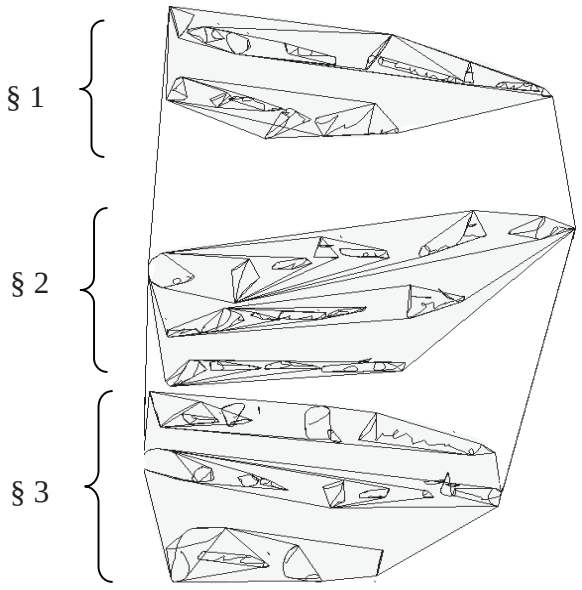
Table 2. Error rate comparison between our approach and a method inspired by the Docstrum for different categories of documents, according to the criteria defined in §3.2.

Table 2 shows the comparison between our adaptation of the Docstrum algorithm and our PFG-based method. Learning of laws parameters was made by cross-validation. We computed 56 experiments taking for each 55 learning documents and one test document. Results from table 2 are average on the 56 experiments. For homogeneous documents, both methods perform well and similarly, about 3.5% error rate for criterion L1. Results are still good when considering reading order for line detection and for lines aggregation into paragraphs. Error rates are however higher for criteria D1 and D2 which concern document level, indicating either sub-segmentation or over-segmentation of pages into paragraphs. For heterogeneous documents, without well defined structure, results are naturally worse but PFG exhibit higher robustness than the Docstrum based methods, Docstrum based method is not efficient in any case here with for example 22% error rate for criteria L1.

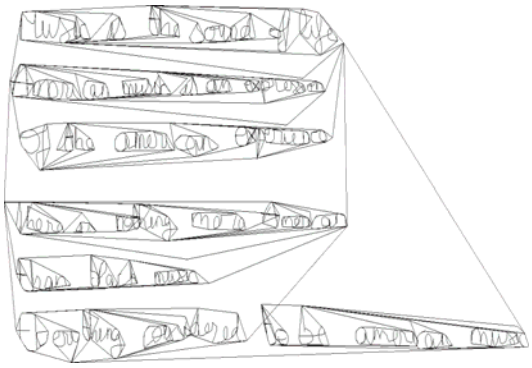
We can see that PFG on heterogeneous documents sometimes provide better rates than PFG on homogeneous documents (L1, D2). In fact, sometimes some features of homogeneous documents can mistake PFG's. For example in Figure 2 d), document is an homogeneous document but PFGs don't find correctly last paragraph because last line doesn't have same features that other lines in second paragraph.



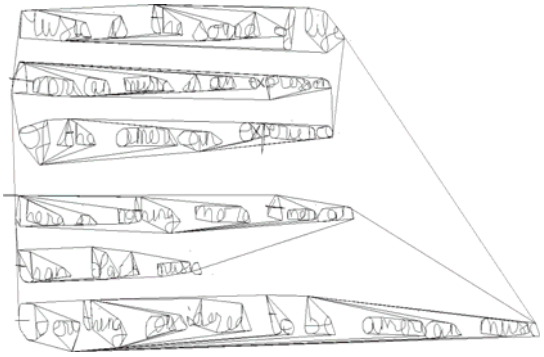
a) Segmentation of an heterogeneous document with 3 paragraphs by Docstrum. The second paragraph contains the first line of the third real paragraph.



b) Segmentation of the same document that in a) by PFGs.



c) Segmentation of an homogeneous document with 2 paragraphs by Docstrum. Three paragraphs are found, the last one corresponds to the end of the last line.



d) Segmentation of the same document that in c) by PFGs. Again, three paragraphs are found but the last one is more understandable and consists in the whole last line which is longer than other lines in the same paragraph.

Figure 2: Segmentations samples

4. Conclusion

The system we presented in this paper is based on an extension of a probabilistic approach for on line handwritten documents segmentation. This method, based on Probabilistic Features Grammars owns some interesting features, it takes context into account and its probabilistic nature allow considering multiple hypotheses. However, those advantages come with a serious algorithmic complexity. We dealt with the huge combinatorial complexity of the task by introducing beam search strategy and by interfacing the parsing algorithm with a genetic algorithm. This genetic algorithm aims at discovering globally consistent sets of terms (e.g. lines) that cover a whole document. Thanks to these improvements, our system may now efficiently deal with standard documents of hundreds of words. We evaluated our method on a home made database containing documents of different qualities. To do this, we defined performances criteria that include reading order information and are thus more adapted to on-line handwritten documents. We validated our work confronting our system to a reference method we adapted from a classical approach of off-line documents segmentation. These preliminary experimental results are promising and show that our system behave well for all kinds of documents while a more classical technique seem more adapted to homogeneous documents only.

5. References

[1] N. Gauthier, T. Artieres, "Poorly Structured Handwritten Document Segmentation Using Probabilistic Feature Grammars", Document Layout Interpretation and its Applications, *Workshop associated to ICDAR*, 2003, pp 375-384.

[2] J. Goodman, "Probabilistic feature grammars", *International Workshop on Parsing Technologies*, 1997, pp 237-264.

[3] K. Jain, A. Namboodiri, J. Subrahmonia, "Structure in on-line documents", ICDAR, 2001, pp 844-848.

[4] K. Kise, A. Sato, M. Iwata, "Segmentation of page images using the area Voronoi diagram", *Computer Vision and Image Understanding*, 70, 1998, pp. 370-382.

[5] U. Marti, H. Bunke, "Text line segmentation and word recognition in a system for general writer independent handwriting recognition", *ICDAR*, 2001, pp 159-163.

[6] G. Nagy, S. Seth, "Hierarchical representation of optically scanned documents", *ICPR*, 1984, pp 347-349.

[7] L. O'Gorman, "The document spectrum for page layout analysis", *IEEE Trans. PAMI*, Vol. 15, 1993, pp. 1162-1173.

[8] E. Ratzlaff, "Inter-line distance estimation and text line extraction for unconstrained online handwriting", *IWFHR*, 2000, pp 33-42.

[9] M. Shilman, Z. Wei, S. Raghupathy, P. Simard, D. Jones, "Discerning Structure from Freeform Handwritten Notes", *ICDAR*, Vol. 1, 2003, pp 60-65.

[10] A. Stolcke, An Efficient Probabilistic "Context-Free Parsing Algorithm that Computes Prefix Probabilities", *Computational Linguistics*, Vol. 21, No. 2, 1995, pp. 165--201.