Analysis of Book Documents' Table of Content Based on Clustering

Liangcai Gao[1], Zhi Tang[1], Xiaofan Lin[2], Xin Tao[1], Yimin Chu[1]
[1]Institute of Computer Science & Technology, Peking University, [2]Vobile Inc
{gaoliangcai, tangzhi, taoxin, chuyimin}@icst.pku.edu.cn ,xiaofan@vobileinc.com

Abstract

Table of contents (TOC) recognition has attracted a great deal of attention in recent years. After reviewing the merits and drawbacks of the existing TOC recognition methods, we have observed that book documents are multi-page documents with intrinsic local format consistency. Based on this finding we introduce an automatic TOC analysis method through clustering. This method first detects the decorative elements in TOC pages. Then it learns a layout model used in the TOC pages through clustering. Finally, it generates TOC entries and extracts their hierarchical structure under the guidance of the model. More specifically, broken lines are taken into account in the method. Experimental results show that this method achieves high accuracy and efficiency. In addition, this method has been successfully applied in a commercial E-book production software package.

1. Introduction

Most multi-page documents have a built-in table of contents (TOC), which is a collection of references to the different components of the document and naturally reflects the logical structure of the entire document [9]. Thus, an efficient approach to multi-page document structure analysis is detecting and analyzing TOC pages, namely TOC recognition, which can satisfy the need to organize the logical units of a document into a systematic structure to facilitate future information retrieval and navigation.

We divide TOC recognition into three sub-tasks: TOC detection, whose goal is to locate the TOC pages in a document; TOC parsing, which segments TOC pages into a number of article references, further segments the article references into logical elements / fields, tags the elements, and extracts the TOC hierarchy; and link determination, which is to detect title pages and link article references on the TOC pages to the title pages.

2. Related works

A number of methods on TOC recognition have been reported in recent years. They focus on different sub-tasks of TOC recognition, as shown in Table 1. According to the features those methods utilize, they can be classified into three types: layout feature-based, POS-based and functional feature-based. And according to the way the models are generated, those approaches can be classified into two types: rule-based and learning-based.

Table 1. Literature summary on TOC recognition

	Method type			
Features used	Layout feature -based		POS-based	functional feature -based
Model generation	Rule -based	Learning-based		
TOC detection	[1,3]			[9,10,11]
TOC parsing	[1,2,3, 4,5]	[6,7]	[8]	[9,10]
Link determination	[4]	[6]		[9, 10,11]

For example, Mandal et al. [1] proposed a method of detecting TOC pages in a document, relying on page number-related heuristics and working on page images. Tsuruoka et al. [2] used the indentation and font size to extract structural elements such as chapters and sections in a book. Luo et al. [3] detected TOC pages through finding the predefined connectors such as dot lines. Lin et al. [4] introduced a system of TOC page analysis using layout modeling and headline matching, and acquired the logical structure of the TOC through in-depth analysis of its numbering scheme. He et al. [5]

combined geometrical rules (indentations) and semantic rules (typical text sequences identifying chapters and sections) to extract the hierarchical structure in Chinese books.

Those rule-based methods are simple and mostly effective. But new rules need to be generated for new document types. With the increase of document types and the generation of numerous rules will becomes a heavy burden. Thus machine learning has been employed to automate the rule or model generation. For example, Satoh et al. [6] used a decision tree to extract bibliographic information, including TOC pages. Bourgeois et al. [7] proposed a probabilistic relaxation method to analyze the structure of periodicals through training on the layouts of several representative samples. Different from those layout-based methods, Belaïd et al. [8] proposed a labeling approach to delimit articles using the contextual rules of part-of-speech (POS) tagging.

There are so many possible layouts for TOC pages in real-world documents that the layout-based methods cannot reliably handle all documents. POS-based methods are language dependent and thus different POS models are required for different languages. To deal with layout variance, some researchers applied functional knowledge to TOC recognition. For example, Lin et al. [9] and Déjean et al. [10] independently used text matching between TOC candidate pages and body pages for detecting TOC pages in a document. Yacoub [11] combined text matching, keywords and page numbers to detect TOC pages. The functional knowledge used in those methods is that the content of a reference in TOC pages will repeat on the title page. Such methods tend to be more robust, but they are usually time-consuming because the computational complexity required by text matching is quadratic to the number of text blocks.

3. Proposed solution

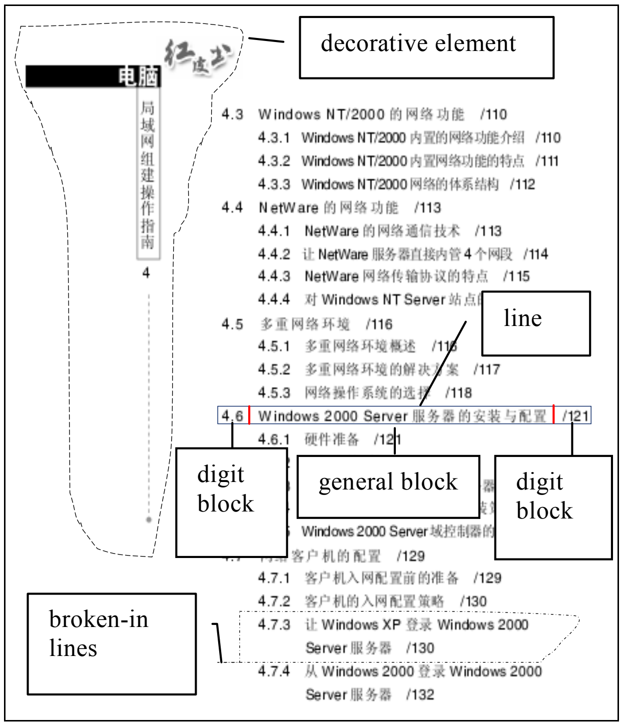
We have observed that document elements belonging to the same component usually share format regularity, although different documents might adopt different formats for a certain type of document components (e.g. headings, table of contents, references, footnotes). We call this "document intrinsic local format consistency". This is a quite generic assumption. For example, the headings at the same level in a book usually adopt a single format. The entries at the same level in TOC pages share the same visual characteristics [10]. The citation strings in the same document also share the same format. So a layout model can be obtained from a subset of elements of a component using statistic features, then the generated model can be used to recognize the remaining elements of the component. This approach is usually effective and applicable from document to document because the "document intrinsic local format consistency" reflects the common typesetting practice and the statistic features can be quickly captured

Under this assumption, a document usually adopts a few formats in most of its TOC entries, which are called dominant TOC formats (referred to as "DTOCF" henceforth). DTOCF serves as the TOC template of a document. Déjean et al. [10] has used this format consistency of TOC to extract the hierarchical structure of TOC. This approach is especially suitable for book documents, which usually have multiple pages and contain a large number of TOC entries. We can generate a TOC model using some of the TOC entries, and then use the model to analyze the other TOC entries. Along this direction, we attempt to solve the problems of TOC parsing and decorative element detection using the DTOCF and clustering. The documents processed in our method are electronically originated PDF files from which the information of fonts, characters and coordinates can be accurately extracted. Section 4 introduces the clustering techniques, Sections 5 and 6 focus on decorative element detection and TOC parsing respectively. Experimental results are discussed in Section 7 and conclusions are drawn in Section 8.

4. Clustering techniques

The decorative elements in TOC pages are usually very different from the dominant TOC content in layout and format, and broken-in TOC lines occur infrequently in TOC pages. So they can be considered as anomalies of TOC content. We select the Microsoft Clustering algorithm [12] as our clustering technology to recognize TOC for its good performance in identifying anomalies from dataset.

The Microsoft Clustering algorithm is a segmentation algorithm provided by Microsoft SQL Server Analysis Services. And it uses iterative techniques to group cases in a dataset into clusters that contain similar samples. After first defining clusters, the Microsoft Clustering algorithm calculates how well the clusters represent the samples, and then tries to refine clusters to better represent the data. The algorithm iterates through this process until it can no longer improve the results. And the Microsoft Clustering algorithm offers two methods for calculating how well points fit within the clusters: Expectation Maximization (EM) and K-Means.



红皮书 电脑 局域网组建操作指南 4

4.3 Windows NT/2000 的网络功能 /110
4.3.1 Windows NT/2000 内置的网络功能介绍 /110
4.3.2 Windows NT/2000 内置网络功能的特点 /111
4.3.3 Windows NT/2000 网络的体系结构 /112
4.4 NetWare 的网络功能 /113
4.4.1 NetWare 的网络通信技术 /113
4.4.2 让 NetWare 服务器直接内管 4 个网段 /114
4.4.3 NetWare 网络传输协议的特点 /115
4.4.4 对 Windows NT Server 站点
4.5 多重网络环境 /116
4.5.1 多重网络环境概述 /116
4.5.2 多重网络环境的解决方案 /117
4.5.3 网络操作系统的选择 /118
4.6 Windows 2000 Server 服务器的安装与配置 /121
4.6.1 硬件准备 /121
Windows 2000 Server 域控制器的
客户机的配置 /129
4.7.1 客户机入网配置前的准备 /129
4.7.2 客户机的入网配置策略 /130
4.7.3 让 Windows XP 登录 Windows 2000 Server 服务器 /130
4.7.4 从 Windows 2000 登录 Windows 2000 Server 服务器 /132

Figure 1. A sample TOC page

5. Decorative element detection

There are usually some redundant elements in TOC pages for decorative purpose, such as headers and footers, vertically-typesetting words surrounding the page or column borders, as shown in Figure 1. And we call them "decorative elements". Those elements can be considered as "noises" in page layout analysis and the other downstream processing of TOC. The previous works on TOC recognition have not considered decorative element detection in TOC pages. In this paper, we use the above-mentioned clustering technology to group the elements in TOC pages. And the decorative elements should be in the anomalies group because they are usually much fewer than the dominant TOC content. Our approach to detect decorative elements includes the following steps:

1) Combine characters into text blocks iteratively. Two characters are combined when their vertical or horizontal distance is below a threshold (e.g. half of font size) and their sizes are similar.
2) Generate a feature vector for each block in all TOC pages. The features include the fonts, the height and width of the bounding box of a block, the number of characters in the block.
3) Cluster with the Microsoft Clustering algorithm on the feature vectors of TOC blocks.
4) Select the clusters with a small number of blocks as anomaly groups. And the blocks in anomaly groups are considered as decorative element candidates.
5) Determine the final decorative elements by the enforcement of some general layout knowledge. For example, decorative elements are usually close to the border of a page and they usually appear in the similar positions throughout the whole TOC pages.

6. TOC parsing

After the decorative element detection, we get the clean TOC text blocks. Then we combine those blocks into TOC entries and extract the hierarchical structure of those entries. Previous works have mentioned very little on how to get the individual TOC entries from TOC pages or how to segment TOC pages into TOC entries. Only Mandal et al. [1] proposed a method to process broken-in lines based on predefined TOC component models, which cannot adapt to various TOC styles. In this paper, we use clustering techniques to generate a matched TOC model based on "document intrinsic format consistency". And the model is employed to parse the document's TOC.

6.1 TOC model generation

The following steps are carried out to generate a TOC model for a horizontally typeset document. Vertically typeset documents can be handled similarly.

1) Iteratively combine blocks into text lines. Two blocks are combined when their horizontal distance is below a threshold (e.g. half of font size), their heights are similar, and they intersect in vertical direction.
2) Detect connectors. The connectors described in [3] are predefined, such as dot lines. Because the symbols in the connectors repeat contiguously, we select characters that repeat over three times contiguously as connector symbol candidates. Then we calculate the number of lines that each connector symbol candidate appears. If the lines in which a connector symbol candidate appears are above a percentage (e.g. 60%) of the total lines, we select the symbol as the final connector symbol.
3) Tag blocks in each line as digit blocks consisting of digits and punctuations, connector blocks consisting of connector, or normal blocks, as shown in Figure 1.

4) Generate a feature vector for each line in all TOC pages. The features include:
   a) The number of blocks contained in a line.
   b) The font of a line. When more than one font is adopted in a line, we select the dominant font as the font of the line.
   c) The height of a line.
   d) The last and first character of a line. They are usually parentheses and some common words in books such as "chapter" and "section".
   e) The tag of each block in a line.
5) Cluster with the Microsoft Clustering algorithm on the feature vectors of TOC lines.
6) Select the clusters with a large number of lines as DTOCF groups and the other groups as anomaly groups, and select the features of a line in the DTOCF groups as the DTOCF features.

6.2 Broken-in lines processing

One TOC entry sometimes contains too much content to finish in one line, so it might have multiple broken-in lines. A broken-in line is different from the regular lines in that it is not a complete TOC entry. Because the broken-in lines occur infrequently in TOC pages, they are usually grouped into anomaly groups after step 5) of Section 6.1. Then we process those lines in the anomaly groups as follows:

1) Select two adjacent lines with similar height from anomaly groups, and combine the later line with the previous line to create a new line $\boldsymbol{p}$.
2) Recalculate the feature vector of $\boldsymbol{p}$, and calculate the distances between the feature vector of $\boldsymbol{p}$ and each DTOCF feature vector. If the smallest distance is below a threshold, we add $\boldsymbol{p}$ into the DTOCF group that has the smallest distance from $\boldsymbol{p}$, and delete the two combined lines. If no distance is below the threshold, the two combined lines are deleted from anomaly groups and $\boldsymbol{p}$ is added as a new anomaly group.
3) Repeat step 1) and 2) until no neighboring lines can be selected.

6.3 Hierarchy determination

Déjean et al. [10] decided the hierarchical organization of the TOC based on the assumption that elements (namely TOC entries) belonging to the same hierarchical level share the same visual characteristics and the elements of the lower level have more adjacent elements from the same level. But the entries with no siblings can potentially lead to incorrect level detection in their method. For example, their method cannot process such cases as shown in Figure 2 because too many nodes at the second level have no siblings. And usually there are more adjacent entries in a Chinese book TOC at the section level than at the subsection level.

We use the general containing relationships of TOC entries' levels to determine the TOC hierarchical structure: the lower level is contained in its adjacent and higher level. After processing broken-in lines, every cluster belongs to a certain level. And we determine the hierarchical relationship between clusters through the following steps:

1) Sort all of the TOC entries in the natural reading order to generate the entry sequence $\boldsymbol{E}$ ($\boldsymbol{e_1}$, $\boldsymbol{e_2}$, … $\boldsymbol{e_i}$…$\boldsymbol{e_{n-1}}$, $\boldsymbol{e_n}$). $\boldsymbol{n}$ is the number of the TOC entries.
2) Assign a level to each entry in $\boldsymbol{E}$ and generate its level sequence $\boldsymbol{L}$ ($\boldsymbol{l_1}$, $\boldsymbol{l_2}$, … $\boldsymbol{l_i}$…$\boldsymbol{l_{n-1}}$, $\boldsymbol{l_n}$). $\boldsymbol{l_i}$ is the level of $\boldsymbol{e_i}$ and is within the range from $\boldsymbol{1}$ to $\boldsymbol{h}$ and $\boldsymbol{h}$ is the total number of the clusters. The entries in the same cluster have the same level and the highest level is assigned $\boldsymbol{1}$. Because a TOC level is always contained in its adjacent and higher level, the increment of the level values of TOC entries in $\boldsymbol{E}$ should be less than $\boldsymbol{2}$. Namely, Level $\boldsymbol{i}$ cannot follow Level $\boldsymbol{i-2}$ or the higher levels. And a regular $\boldsymbol{L}$ should conform to the following condition: the difference of any two neighboring elements of $\boldsymbol{L}$ is below $\boldsymbol{2}$. For example, the $\boldsymbol{L}$ corresponding to Figure 2 is (1,2,3,3,1,2,3,2,2,2,2,2). Thus, we can find the hierarchical relationship between clusters through assigning the levels to clusters so that the corresponding $\boldsymbol{L}$ satisfies that condition. The level values of clusters are initialized based on some heuristics. For example, the cluster with a larger font or the cluster containing fewer TOC entries should belong to the higher level and thus have smaller level values.
3) Check the level sequence $\boldsymbol{L}$ of $\boldsymbol{E}$. If it satisfies the above condition, we finalize the clusters' levels described in step 2). Otherwise, we randomly assign the clusters' levels between $\boldsymbol{1}$ and $\boldsymbol{h}$, and repeat Steps 2 and 3.

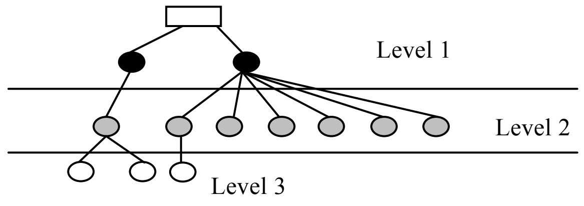


Figure 2. Tree structure of a TOC: the reading order is decided through depth-first traversal

7. Experimental results

We select 150 Chinese electronic books with various TOC styles. The 150 books contain 734 TOC pages, 11478 TOC entries and 247 broken-in lines in total.

We measure the precision of TOC decorative element detection, TOC entry generation, TOC hierarchy determination, and broken-in line combination. The test results are show in Table 2. And the average execution time for a book is around 3 seconds on a personal computer (2GHz CPU, 512MB RAM).

Table 2. Experimental results

	Precision	Time (ms)
Decorative data detection	97.2	1126
Entry generation	93.7	1785
Hierarchy determination	98.4	134
Broken-in line processing	98.3	68

We have also analyzed the errors and here is a summary:

- Decorative element detection: Some books have background text in TOC pages. And the text usually has an enough number so that it is considered as regular text.
- TOC entry segmentation: We assume that the most TOC entries are single lines, which is the most common case in Chinese book. So the method fails when a book's TOC entries mostly appear in multiple lines.
- TOC hierarchy extraction: Some books have the same format across two different levels. As a result our method groups the entries in the two levels into a cluster.

In addition, it is noteworthy that our TOC recognition approach has been integrated with a commercial Chinese electronic book (E-Book) publishing software package. After being extracted automatically by the proposed TOC recognition approach and modified manually, the TOCs with target links are embedded in E-Books. And about 10,000 books have actually gone through the software in the last past nine months.

8. Conclusions

In this paper, we describe the methods to automatically detect decorative elements from TOC pages, to generate TOC entries, and to extract TOC hierarchical structure. The preliminary experiments demonstrate the effectiveness of those methods. The most significant differentiator of our methods is that they rely on a very generic assumption: document elements belonging to the same component will share some formal regularity, namely document intrinsic local format consistency. So our methods can be applied to a wide range of documents and still enjoy the efficiency of layout-based methods.

9. References

[1] S. Mandal, S.P. Chowdhury, A.K. Das, and B. Chanda, "Automated Detection and Segmentation of Table of Contents Page from Document Images", *Proc. of the ICDAR'03*, IEEE Computer Society Press, Edinburgh, 2003, pp. 398-402.

[2] S. Tsuruoka, C. Hirano, T. Yoshikawa, and T. Shinogi, "Image-based Structure Analysis for a Table of Contents and Conversion to XML Documents", *Proc. of the DLIA '01*, Seattle, 2001.

[3] Q. Luo, T. Watanabel, and T. Nakayama, "Identifying Contents Page of Documents", *Proc. of the ICPR'96*, IEEE Computer Society Press, Vienna, 1996, pp. 696-700.

[4] C. Lin, Y. Niwa, and S. Narita, "Logical Structure Analysis of Book Document Images Using Contents Information", *Proc. of the ICDAR'97*, Ulm, 1997, pp. 1048-1054.

[5] F. He, X. Q. Ding, and L. R. Peng, "Hierarchical Logical Structure Extraction of Book Documents by Analyzing Tables of Contents", *Proc. of SPIE Conference on Document Recognition and Retrieval IX*, San Jose, 2004, pp. 6-13.

[6] S. Satoh, A. Takasu, and E. Katsura, "An Automated Generation of Electronic Library Based on Document Image Understanding", *Proc. of the ICDAR'95*, Montreal, 1995, pp. 163-166.

[7] F. L. Bourgeois, H. Emptoz, and S. S. Bensafi, "Document Understanding Using Probabilistic Relaxation: Application on Tables of Contents of Periodicals", *Proc. of the ICDAR'01*, IEEE Computer Society Press, Seattle, 2001, pp. 508-512.

[8] A. Belaïd. "Recognition of Table of Contents for Electronic Library Consulting". *International Journal on Document Analysis and Recognition (IJDAR)*, Berlin, 2001, 4(1), pp. 35-45.

[9] X. F. Lin, and Y. Xiong, "Detection and Analysis of Table of Contents Based on Content Association", *International Journal on Document Analysis and Recognition (IJDAR)*, Berlin, 2006, 8(2-3), pp. 132-143.

[10] H. Déjean, and J. L. Meunier, "Structuring Documents According to Their Table of Contents", *Proc. of Symposium on DocEng'05*, ACM, Bristol, 2005, pp. 2-9.

[11] S. Yacoub, and J. A. Peiro, "Identification of Document Structure and Table of Content in Magazine Archives", *Proc. of the ICDAR'05*, IEEE Computer Society Press, Seoul, 2005, pp. 1253-1257.

[12] http://technet.microsoft.com/zh-cn/library/ms174879.aspx