Computational Linguistics and Chinese Language Processing
Vol. 13, No. 4, December 2008, pp. 443-460

Automatic Wikibook Prototyping via Mining Wikipedia

Jen-Liang Chou[*], and Shih-Hung Wu[*]

Abstract

Wikipedia is the world’s largest collaboratively edited source of encyclopedic knowledge. Wikibook is a sub-project of Wikipedia that is intended to create a book that can be edited by various contributors, similar to how Wikipedia is composed and edited. Editing a book, however, requires more effort than editing separate articles. Therefore, methods of quickly prototyping a book is a new research issue. In this paper, we investigate how to automatically extract content from Wikipedia and generate a prototype of a Wikibook as a start point for further editing. Applying search technology, our system can retrieve relevant articles from Wikipedia. A table of contents is built automatically and is based on a two-stage searching method. Our experiments show that, given a keyword as the title of a book, our system can generate a table of contents, which can be treated as a prototype of a Wikibook. Such a system can help free textbook editing. We propose an evaluation method based on the comparison of system results to a traditional textbook and show the coverage of our system.

Keywords: Wikipedia, Wikibook, Table of Contents Generation

1. Introduction

The ability to quickly construct a free encyclopedia, such as Wikipedia, has shown that the Web 2.0 has been successful. Community and interactivity among users on the Internet has become a popular topic. A project named “*Science Online,*” which brought the wiki scheme to schools, lets students participate in collaborative writing (Forte & Bruckman, 2007). Wikipedia is useful in college education, both for general topics (Lally & Dunford, 2007, May/June) and for specific topics, such as physics (Muchnik, Itzhack, Solomon, & Louzoun, 2007). In this paper, we focus on another project of the Wikimedia Foundation, Wikibook, which is also useful in the classroom (Sajjapanroj, Bonk, Lee, & Lin, 2006). Wikibook provides free textbooks on the Internet via the Wiki system, letting global users edit the

[*] Department of Computer Science and Information Engineering, Chaoyang University of Technology, Taiwan R.O.C.
E-mail :shwu@cyut.edu.tw

[Received September 15, 2008; Revised Mar. 19, 2009; Accepted Mar. 25, 2009]

contents of textbooks. Creating a book without supporting data, however, is difficult. An expert or a system that can provide a general framework and useful references for a book is of much help. Thus, we propose an automatic Wikibook prototyping system that can pick relevant articles from Wikipedia and construct a hierarchy as the table of contents for a given topic. Our system consists of information retrieval and web mining technology.

In previous work, the TOC and anchor text of a Wikipedia entry has been used to form the TOC of a Wikibook (Yang, Han, Oh, & Kwak, 2007). A relevant research area is Topic Maps. Topic Maps are analogous to the Table of Contents (TOC) of a textbook. Users can realize and memorize the relevant concepts of a topic. “Topic Maps for learning” (TM4L) (Dicheva & Dichev, 2005) is an application of Topic Maps. These methods, though, mostly rely on humans without using information retrieval technology, which can provide a considerable amount of relevant information. Studies in knowledge acquisition provide some hints. Semi-automatic methods can aggregate a domain ontology via Internet search (Roberson & Dicheva, 2007).

We propose a method that can help prototype a Wikibook automatically using the contents from Wikipedia. In the following sections, we will describe our methodology in Section 2, system implementation in Section 3, experimental results in Section 4, discussions in Section 5, and give conclusion in the final section.

2. Methodology

We propose a framework for Wikibook prototyping which involves several modules. These modules can be replaced to fulfill the needs of different requirements. For example, we might customize the system for different languages, users of different ages, or topics with different contexts.

2.1 Corpus Preparation

The first module is the preparation of a corpus. We can use the whole of Wikipedia, certain language versions, or subsets as the searching target. The system then extracts and analyzes contents from the corpus. As a knowledge source, Wikipedia provides not only the content but also a lot of links to contexts, which are also valuable. We will extract keywords from the content of Wikipedia pages, and will find related terms from the anchor text in these pages.

2.2 Search Engine and Anchor Text Miner

The second module is a search engine and an anchor text miner. As mentioned above, relevant topics can be found not only from a full text keyword search, but also from links in Wikipedia. This module is important from the technical point of view. Our system searches relevant topics and their hyponyms using information retrieval technology. On the other hand, the anchor text miner can extract related terms from the anchor texts of the retrieved pages.

2.3 Hierarchical Construction

Given relevant topics, our system then generates a hierarchy. This hierarchy can be viewed as the table of contents of a Wikibook for further editing.

3. Implementation issues

Our methodology gives a general idea on how to generate a Wikibook automatically within a flexible framework. In the following sections, we discuss our system and experiments on computer science topics in both the English and Chinese versions of Wikipedia. Figure 1 shows the architecture of our system.

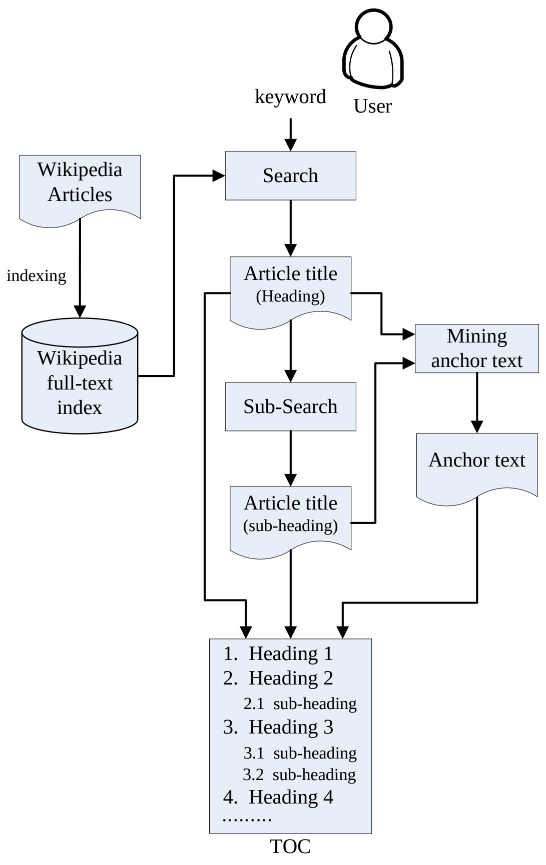


Figure 1. System Architecture

3.1 Corpus Choice

We chose Wikipedia as our corpus in the experiments for two reasons. First, the idea of the availability of the corpus; it is an ample resource of information which is available for every potential Wikibook editor and there are no copyright issues in regards to using the content for a Wikibook. Second, the quality of content: there is more professional knowledge in Wikipedia than in general Websites, and potential editors can compile the content from Wikipedia with less effort.

3.2 Search Strategy

Our search strategy is automatic two-stage iterative searching. The system takes a term as the query and performs context searching via a standard information retrieval process. We use pseudo-relevance feedback as our searching algorithm. The resulting set of the first search will be used as the corpus in the second stage.

The system ranks the search results according to a traditional TF-IDF scoring function that is restated in Formula (1) where $Score_i$ is the ranking score of an article, i denotes an article, j denotes a term occurring within the query, and T denotes the number of terms in the query. TF_j is the frequency of j occurring within the article title, TF_{ij} is the frequency that term j occurs in article i. We assume $token_{ij} = \left| \left\{ T_i - j \right\} \right|$ as the number of the rest words after deleting term j from the article i, where T_i is the number of terms in the article i:

$$Score_i = \sum_j^T \left(\frac{TF_j \times IDF_j}{\sqrt{\sum_j \left(TF_j \times IDF_j \right)^2}} \right) \times \left(\frac{TF_{ij} \times IDF_j}{\sqrt{token_j}} \right) \quad (1)$$

where $IDF_j = \log\left(\frac{|D|}{|D_j|} + 1 \right)$, measures term j involvement in other documents, D is all the articles in the index, D_j is articles that contain j.

Table 1 shows a collection of four documents, and the corresponding value of each variable in Formula (1) is shown in Table 2. Suppose the query terms are “A B C”, then T is 3, and i can be 1 to 4 as the Doc ID in Table 1. Then, the Score of a document according to the formula is shown in the last column of Table 2. Where $Score_1 = 0.551$; $Score_2 = 0.260$; $Score_3 = 0.675$; $Score_4 = 0$, we rank these scores from high to low. From this, we can attain results such that, if we input the query “A B C”, the system will output the order of the document as 3, 1, 2, 4.

Table 1. Example Documents

Doc ID	Title	Contents
1	A	A B A E C
2	B	B D F
3	C	C A C E
4	D	D E F B

Table 2. Calculate the value of each notion using Formula (1)

i	j	TF_j	TF_{ij}	$\|D\|$	$\|D_j\|$	IDF_j	$token_{ij}$	$Score_i$
1	A	1	2	4	2	0.477	3	0.551
1	B	0	1	4	3	0.368	4	0
1	C	0	1	4	2	0.477	4	0
2	A	0	0	4	2	0.477	3	0
2	B	1	1	4	3	0.368	2	0.260
2	C	0	0	4	2	0.477	3	0
3	A	0	1	4	2	0.477	3	0
3	B	0	0	4	3	0.368	4	0
3	C	1	2	4	2	0.477	2	0.675
4	A	0	0	4	2	0.477	4	0
4	B	0	1	4	3	0.368	3	0
4	C	0	0	4	2	0.477	4	0

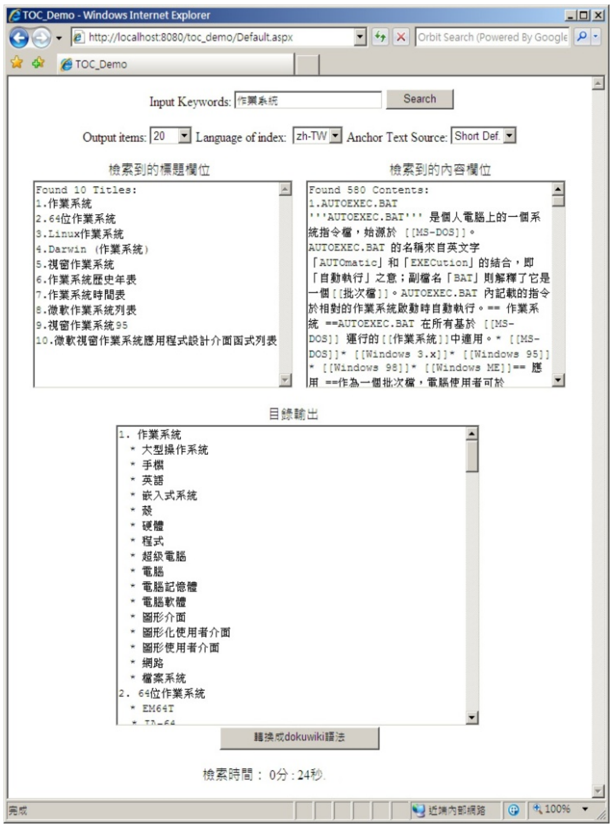
This means the frequency of a term occurring both in the title and in the article is important. The higher the term frequency is, the higher the score of the article will be.

After filtering out the noise in the resulting set, such as redirected pages, our system maintains the top N documents, which are the highest score articles, as the resulting set for further search and the user can customize the arbitrary parameter N. Our system thus finds the first level of relevant topics from the resulting set. These relevant topics can be treated as the backbone of a Wikibook.

3.3 Sub-Topic Finding

Since the extracted topics from the first search often contain the original query term, our system removes the original query term string and uses the reduced topics as the query terms in the second stage search based on the result of the first stage search. For example, if we input the keyword “Operating system (作業系統)” and retrieve the topic “Linux Operating System (Linux 作業系統)”, the shortened query will be “Linux”. As another example, if we input the keyword “Operating system (作業系統)” and retrieve the topic “Windows Operating System (視窗作業系統),” the shortened query will be “Windows (視窗)”. This query reformulation is the same both in the English version and the Chinese version.

Our system extracts keywords from the second stage search result as the sub-level topics of the output TOC. This is a recursive method; we can find the sub-sub-topics in the same manner. For example, we can further search for sub-topics of “Linux” or “Windows”. After finding the sub-level topics, our system will extract other related terms from the anchor text in these Wikipedia articles. As such, every related topic we find corresponds to a related Wikipedia article, which might be useful content of a Wikibook. There are two approaches for mining the related terms: one is to extract anchor texts only in the short definition of an article; the other is to extract from the whole article. Our system then combines these related terms as the sub-level topics.


TOC_Demo - Windows Internet Explorer
http://localhost:8080/toc_demo/Default.aspx
Orbit Search (Powered By Google
TOC_Demo
Search
轉換成dokuwiki語法
完成
近端內部網路
100%

Input Keywords: 作業系統

Output items: 20 Language of index: zh-TW Anchor Text Source: Short Def.

檢索到的標題欄位

```
Found 10 Titles:
1.作業系統
2.64位作業系統
3.Linux作業系統
4.Darwin (作業系統)
5.視窗作業系統
6.作業系統歷史年表
7.作業系統時間表
8.微軟作業系統列表
9.視窗作業系統95
10.微軟視窗作業系統應用程式設計介面函式列表
```

檢索到的內容欄位

```
Found 580 Contents:
1.AUTOEXEC.BAT
'''AUTOEXEC.BAT''' 是個人電腦上的一個系
統指令檔，始源於 [[MS-DOS]]。
AUTOEXEC.BAT 的名稱來自英文字
「AUTOmatic」和「EXECution」的結合，即
「自動執行」之意；副檔名「BAT」則解釋了它是
一個[[批次檔]]。AUTOEXEC.BAT 內記載的指令
於相對的作業系統啟動時自動執行。== 作業系
統 ==AUTOEXEC.BAT 在所有基於 [[MS-
DOS]] 運行的[[作業系統]]中適用。* [[MS-
DOS]]* [[Windows 3.x]]* [[Windows 95]]
* [[Windows 98]]* [[Windows ME]]== 應
用 ==作為一個批次檔，電腦使用者可於
```

目錄輸出

```
1. 作業系統
  * 大型操作系統
  * 手機
  * 英語
  * 嵌入式系統
  * 殼
  * 硬體
  * 程式
  * 超級電腦
  * 電腦
  * 電腦記憶體
  * 電腦軟體
  * 圖形介面
  * 圖形化使用者介面
  * 圖形使用者介面
  * 網路
  * 檔案系統
2. 64位作業系統
  * EM64T
  * IA-64
```

檢索時間： 0分 : 24秒.

Figure 2. User Interface

4. Experiment

4.1 Dataset Preparation

We downloaded the English and Chinese version of Wikipedia dump data from the Wikimedia downloads website (http://download.wikimedia.org/enwiki/). Then, we parsed the content and stored it in a MySQL database. In the following experiment, our system uses the data under the title and content XML tags. A searching tool is built based on the Lucene open source information retrieval API (http://lucene.apache.org/). The user interface is shown in Figure 2.

Our system can generate a TOC for any given title without other information. To evaluate the quality of our system, we conducted several experiments to find differences with previous work. As the target of comparison, we first observe a manually edited Wikibook entitled “Operating System Design”; part of the TOC is shown in Table 3. The first four level one topics in this TOC are general concepts, which are independent of the title of the Wikibook. The sub-topics of and after the fifth topic, however, are very likely to have been generated automatically. We also find that the sub-topic of the fifth topic, “Kernel Architecture,” states the concept of “Kernel Architecture” or gives some illustrations. Our system can help to automatically generate this part of a Wikibook TOC.

Table 3. Partial view of the Wikibook TOC of “Operating System Design”, which was edited manually

```
1. Preface
2. Introduction
3. Case studies
4. History
5. Kernel Architecture
   5.1 Monolithic Kernels
      5.1.1 Solaris
      5.1.2 Linux
      5.1.3 Windows 9x
   5.2Microkernels
      5.2.1 QNX
   5.3Exokernel
      5.3.1 XOK
   5.4 Hybrid Kernels
      5.4.1 Windows NT/XP
      5.4.2 Mac OSX
      5.4.3 BeOS
6. Initialisation
   6.1 Boot Loaders
   6.2 Hardware Initialisation
7. Processes
…
```

4.2 Experiment 1: Generate the TOC of “Operating System Design” via our System

As the first example, to derive a TOC about operating systems, we input the query term “Operating system,” into our system. The system automatically generates a corresponding TOC, shown in Table 4. The arbitrary N can be 10, 20, 50, 100, or 1000. In our experiment, we choose the Top 20 (N=20) as our output number. The user can choose different N for greater or fewer topics in our system. The sub-topics are ranked according to Formula (1), discussed in Section 3.2. Next, the system uses the scheme described in Section 3.3 to perform a second stage search. In this case, the sub-topics at the second level are adequate. For example, the sub-topics of “Linux operating system” in Table 4 are instances of the Linux operating system. The scheme in Section 2.3 is effective.

Table 4. Partial view of automatically generated TOC of “Operating System”

1. Operating system
2. THE operating system
3. IBM AIX (operating system)
4. CPM operating system
5. Disk operating system
6. Kent Applicative Operating System
7. Linux operating system
    7.1 Distro
    7.2 Flask operating system
    7.3 Sabayon
    7.4 Red hatter
8. Operating system advocacy
9. Real-time operating system
    9.1 Rubus (disambiguation)
    9.2 RTMOS (Real-Time Multiprogramming Operating System)
10. VSE (operating system)
    10.1 Exec
    10.2 Protected procedure call
11. Computer network operating system
    11.1 Faceless process
12. Network operating system
    12.1 Faceless process
    12.2 Brazil (operating system)
13. Solaris (operating system)
14. Pick Operating System
15. Darwin (operating system)
16. Operating system/kernel
    16.1 Flask operating system

Table 5. Partial view of automatically generated TOC of the Chinese version “作業系統(Operating System)”

1.作業系統
2.64位作業系統
3.Linux作業系統
    3-1.Ebuntu
4.Darwin (作業系統)
5.視窗作業系統
    5-1.AUTOEXEC.BAT
    5-2.JavaOS
    5-3.Xfwm
    5-4.WINGs Display Manager
6.作業系統歷史年表
7.作業系統時間表
8.微軟作業系統列表
9.視窗作業系統95
10.微軟視窗作業系統應用程式設計介面函式列表

Table 5 shows the result of the same experiment in the Chinese version of Wikipedia. The result is similar but with less recall. Since the size of the Chinese version of Wikipedia is much smaller than the English version of Wikipedia, this is a reasonable result. Table 6 shows the results of searching and mining the anchor text, which is described in Section 2.3. As we can see, the topic has * notion, meaning it is the anchor text, and the results show more topic information differences with Table 3.

Table 6. Partial view of automatically generated TOC of the Chinese version "Operating System", plus mining anchor texts from short definition

```
1.作業系統
        * 大型操作系統
        * 手機
        * 英語
        * 記憶體
        * 嵌入式系統
        * 殼
        * 硬體
…
2.64 位作業系統
        * EM64T
        * IA-64
        * Linux
        * Windows
…
3.Linux 作業系統
        * 386
        * Apache
        * DEV C++
        * GNOME
        * GNU 工程
…
    3-1.Ebuntu
        * 12 月 4 日
        * 2006 年
        * Edubuntu
…
4.Darwin (作業系統)
        * 2000 年
        * 2002 年 4 月
        * 2003 年 7 月
        * 2005 年 5 月
…
```

4.3 Experiment 2: Using the crawling method to generate the TOC of "Operating System Design"

We perform the same experiment with the crawling method described in the literature (Yang, Han, Oh, & Kwak, 2007), which takes a full TOC of a manually edited Wikipedia entry as the input, and outputs a more detailed TOC. The partial result is shown in Tables 7 and 8.

Table 7. Partial result of the TOC generated by the crawling method for the English version of "operating system"

```
1 Technology
      1.1 Program execution
         * Kernel
         * Process(computing)
      1.2 Interrupts
         * interrupt
         * kernel
         * register
         * stack
…
2 Security
         * classified information
         * emulates
         * file transfer protocols
         * firewalls
         * Government Department of Defense
         * p-code
         * Popek and Geldberg virtualization requirment
         * sandbox
         * Trusted Computer System Evaluation Criteria
      2.1 Example: Microsoft Windows
         * access privileges
         * administrator
      …
3 File system support in modern operating systems
      3.1 Linux and UNIX
         * ext2
         * ext3
         * FAT
         * GFS
         * GFS2
         * HFS
         * ISO 9660
     …
```

```
4 Graphical user interfaces
        * CDE
        * context switch
        * COSE
        * GNOME
        * graphical user interface
     …
5 History
        * 80286
        * AmigaOS
        * batch processing
        * Control Data Corporation
    …
6 Mainframes
        * ALGOL
        * B5000
        * Burroughs Corporation
     …
```

Table8. Partial result of the TOC generated by the crawling method for the Chinese version of "operating system"

```
1 歷史
      * 工作排序
      * 分時機制
      * 分散式系統
      * 批次樣式
   1.1 1980 年代前
      * 1947 年
      * 1963 年
      * 1964 年
      * 1970 年代
      * AT&T
      * C 語言
      * Direct access storage device
      * IBM System/360
      * Maurice V. Wilkes
      * Multics
      * Multics
      * Unix
      * 大型電腦
…
```

4.4 Evaluation by Comparing to Traditional Textbook

To measure the coverage of our system results over a traditional textbook, we compare the generated TOC to the TOC of a traditional textbook and show the hits and the precision. Table 9 is a partial TOC adopted from a textbook “Operating System Concepts” (Silberschatz, Galvin, & Gagne, 2001). A traditional textbook provides a suitable coverage, such that we can estimate how much content can be put into one book.

As in the TOC of the current Wikibook in Table 3, this TOC begins with general topics which can be used for each book. Then, it follows the main ideas, which are the most important topics. The order of the topics might be different according how the authors rank them. Nevertheless, there should be some causality, from problem to solution or from general idea to special case. In order to achieve such a goal (finding more relationships between topics (Völkel, Krötzsch, Vrandecic, Haller, & Studer, 2006), more AI technology might be involved, such as frame or logical inference. A predefined frame might guide a system to search general information for a Wikibook. Logic can be used to infer the causality between topics.

Table 9. TOC of an “Operating System” Textbook

1	Introduction
	1.1 What Is an Operating System?
	1.2 Mainframe Systems
…	
2	Computer-System Structures
	2.1 Computer-System Operation
…	
3	Operating-System Structures
	3.1 System Components
…	
4	Processes
	4.1 Process Concept
…	
Chapter 5	Threads …

We separate the number of matching hits and precision rate of the first level and second level in the following table. Table 10 shows the result using the English version. The number of first level and second level topics in the textbook is 23 and 177, respectively. Table 11 shows the results using the Chinese version. The number of first level and second level topics in the textbook is 22 and 191, respectively. We adopt two matching criteria: rigid for exact matching and relaxed for partial matching. Table 12 shows the evaluation results of English version using the crawling method, and Table 13 shows that of the Chinese version.

In this paper, we perform three experiments on our system. Searching only method cannot provide the user more precise concepts because of the low number of hits. The searching method plus mining short definitions, though, can give more hits than the above and attain better precision. On the other hand, the searching method plus mining full document definitions can give better hits, but lose some precision. The result is very similar to the result of crawling method, which requires a manually edited TOC as input.

Comparing our method plus mining anchor text with crawling method, we discuss two directions. In the first level, we can gain the same number of hits. In the second level, if the user wants fewer sub-topics for further editing, mining short definitions can archive precision close to crawling method. On the other hand, if the user wants to get more relative subtopics, mining full document definitions attains more hits than crawling method. In the Chinese version, we can get the same results.

Table 10. Hits and precision of the TOC of "Operating system"

	Searching method			Searching plus mining Short definition			Searching plus mining full document definition		
	# of output	Hits	Precision	# of output	Hits (Short)	Precision	# of output	Hits (Long)	Precision
First Level	20	**5**	0.250						
Second Level (Relax)	17	3	0.176	340	68	**0.200**	2154	**284**	0.132
Second Level (Rigid)		2	0.118		36	**0.106**		**93**	0.043

Table 11. Hits and precision of the TOC of Chinese version "Operating system"

	Searching method			Searching plus mining Short definition			Searching plus mining full document definition		
	# of output	Hits	Precision	# of output	Hits (Short)	Precision	# of output	Hits (Long)	Precision
First Level	10	2	0.200						
Second Level (Relax)	5	1	0.200	179	43	**0.240**	1273	**150**	0.118
Second Level (Rigid)		0	0.000		21	**0.117**		**61**	0.048

Table 12. Hits and precision of the TOC of "Operating system" using crawling method

	Searching method [18]		
	# of output	Hits	Precision
First Level	13	**5**	0.385
Second Level (Relax)	406	105	0.259
Second Level (Rigid)		46	0.113

Table 13. Hits and precision of the TOC of Chinese version "Operating system" using crawling method

	Searching method [18]		
	# of output	Hits	Precision
First Level	8	**3**	0.375
Second Level (Relax)	280	54	0.193
Second Level (Rigid)		30	0.107

5. Discussion and Future Work

According to the observations in the previous section, we would like to discuss issues that might improve the results of automatic Wikibook prototyping.

5.1 Fast Prototyping of Wikibooks

Speeding up editing collaboratively is our goal. Our system can generate a prototype of a Wikibook on any topic. We present our system to depict the TOC in relevant concepts of a book, and it looks like the TOC in general books. It will let editors get relevant concepts of the given topic more quickly and with a good starting point. The TOC generated by our system can be a prototype for editors to revise. Thus, it saves time and stimulates interest in editors to revise it. Since the topics are related articles in Wikipedia, they also provide contents of that topic. Thus, our system not only provides TOC but also some related content.

5.2 Identification of Hypernym/Hyponym Relation

To know the relations between relevant topics is very important in this application, especially the relation between the upper and lower concepts, known as the hypernym/hyponym relation.

With this relation, a tree structure can be built and a hierarchy formed (Nguyen, Matsuo, & Ishizuka, 2007). Supervised (Aggarwal, Gates, & Yu, 1999) or semi-supervised methods (Huang, Zhang, & Lam, 2006) of hierarchical clustering algorithms are promising methods. After finding a set of relevant documents, a system may be clustered into a hierarchy according to the content. The titles of these articles can be treated as the TOC. A knowledge-based approach is also possible. We can try to identify the hypernym/hyponym relation between relevant titles by using WordNet (Farreres, Rodríguez, & Gibert, 2002) or SUMO.

5.3 Importance of an Article

Currently, our system ranks relevant documents according to the scoring function of Lucene. We might combine the result with Google's PageRank (Page, Brin, Motwani, & Winograd, 1999). Each page of the Wikipedia entry contains a link to a page that reports how many entries link to this entry, *cf.* Figure 3. The more inward the link is, the higher the importance of this entry is. With the analysis of the link relation, importance can be ranked (Wissner-Gross, 2006). This information helps to decide whether the TOC should contain this entry or not. The relatedness of words in Wikipedia might also help (Ponzetto & Strube, 2007). In the future, we will acquire more TOC of good Wikibooks as a training set to develop a better process via machine learning algorithm. Clustering algorithms are promising tools for this task.

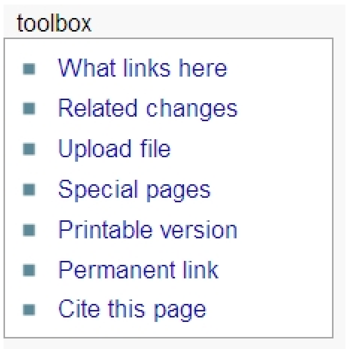


Figure 3. "What links here" in each page of Wikipedia entry links to a page that reports all the entries that link to this entry

6. Conclusion

We proposed an automatic process that can generate a Wikibook by mining the content of Wikipedia. Our method involves document searching, keyword extraction, related term mining, and hierarchy construction technology. We built an experimental system and conducted primary experiments using both English and Chinese. The results showed the automatically generated TOC might help the community to edit a Wikibook more rapidly.

Previous works have not evaluated their system. In this paper, we proposed a method to evaluate the result by comparing to the traditional textbook and report the hits and precision. This gives a standard to compare systems. We find that using the anchor text from the short definition or the full article of Wikipedia will give better precision and more useful terms. The major improvement over previous work is that we do not need a manually edited TOC. Our system uses a searching mechanism that requires only the title as a search keyword, rather than using the full TOC in a manually edited article.

Acknowledgement

This research was partly supported by the National Science Council under NSC 96-2221-E-324-046.

References

Aggarwal, C. C., Gates, S. C., & Yu, P. S. (1999). On the merits of building categorization systems by supervised clustering. In *Proceedings of the Fifth ACM SIGKDD international Conference on Knowledge Discovery and Data Mining*. KDD '99., 352-356.

The Apache Software Foundation. (2008). Lucene Project. Retrieved December 26, 2007, http://lucene.apache.org/

Dicheva, D., & Dichev, C. (2005). Authoring educational topic maps: can we make it easier? In *Proceedings of the Fifth IEEE International Conference on Advanced Learning Technologies*. ICALT'06., 216-218.

Forte, A., & Bruckman, A. (2007). Constructing text:: Wiki as a toolkit for (collaborative?) learning. In *Proceedings of the 2007 international Symposium on Wikis*. WikiSym '07., 31-42.

Farreres, J., Rodríguez, H., & Gibert, K. (2002). Semiautomatic creation of taxonomies. In *Coling-02 on Semanet: Building and Using Semantic Networks - Volume 11* International Conference On Computational Linguistics., 1-7.

Huang, R., Zhang, Z., & Lam, W. (2006). Refining hierarchical taxonomy structure via semi-supervised learning. In *Proceedings of the 29th Annual international ACM SIGIR Conference on Research and Development in information Retrieval*. SIGIR '06., 653-654.

Lally, A. M., & Dunford, C. E. (2007, May/June). Using Wikipedia to Extend Digital Collections. *D-Lib Magazine*, 13.

Muchnik, L., Itzhack, R., Solomon, S., & Louzoun, Y. (2007). Self-emergence of knowledge trees: Extraction of the Wikipedia hierarchies. *The American Physical Society*, Phys. Rev. E 76, 016106.

Nguyen, Dat P.T., Matsuo, Y., & Ishizuka, M. (2007). Subtree Mining for Relation Extraction from Wikipedia. In *Proceeding of NAACL/HLT 2007*, Companion Volume, 125-128.

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. *Technical Report*, Stanford University.

Ponzetto, S. P., & Strube, M. (2007). An API for Measuring the Relatedness of Words in Wikipedia. *Proceedings of the ACL 2007 Demo and Poster Sessions*, 49-52.

Roberson, S., & Dicheva, D. (2007). Semi-automatic ontology extraction to create *draft* topic maps. In *Proceedings of the 45th Annual Southeast Regional Conference*. ACM-SE 45., 100-105.

Sajjapanroj, S., Bonk, C., Lee, M., & Lin, G. (2006). The Challenges and Successes of Wikibookian Experts and Want-To-Bees. In T. Reeves & S. Yamashita (Eds.), *Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education* 2006, 2329-2333.

Silberschatz, A., Galvin, P. B., & Gagne, G. (2001). *Operating System Concepts, Sixth Edition*. John Wiley & Sons.

Völkel, M., Krötzsch, M., Vrandecic, D., Haller, H., & Studer, R. (2006). Semantic Wikipedia. In *Proceedings of the 15th international Conference on World Wide Web*. WWW '06., 585-594.

Wissner-Gross, A. D. (2006). Preparation of Topical Reading Lists from the Link Structure of Wikipedia. In *Proceedings of the Sixth IEEE international Conference on Advanced Learning Technologies*. ICALT'05., 825-829.

Wikimedia Foundation. (2008). Wikimedia Downloads / enwiki. Retrieved December 26, 2007, http://download.wikimedia.org/enwiki/

Yang, J., Han, J., Oh, I., & Kwak, M. (2007). Using Wikipedia technology for topic maps design. In *Proceedings of the 45th Annual Southeast Regional Conference*. ACM-SE 45, 106-110.