

Can Back-of-the-Book Indexes be Automatically Created?

Zhaohui Wu[†], Zhenhui Li[‡], Prasenjit Mitra^{†‡}, C. Lee Giles^{†‡}

[†]Computer Science and Engineering, [‡]Information Sciences and Technology
Pennsylvania State University, University Park, PA 16802, USA
zzw109@psu.edu, {jessieli, pmitra, giles}@ist.psu.edu

ABSTRACT

Automatic creation of back-of-the-book indexes remains one of the few manual tasks related to publishing. Inspired by how human indexers work on back-of-the-book indexes creation, we present a new domain-independent, corpus-free and training-free automation approach. Given a book, the index terms will be sequentially selected according to an indexability score encoded by the structure information residing in a book as well as a novel context-aware term informativeness measurement utilizing the power of the web knowledge base such as Wikipedia. By extensive experiments on books from various domains, we show our approach to be a more effective and practical than ones that used previous keyword extraction and supervised learning.

Categories and Subject Descriptors

H.3.3 [Information Search and Retrieval]: Text Mining;

H.4 [Information Systems Applications]: Miscellaneous

Keywords

Back-of-the-Book Index, Book Index, Term Informativeness

1. INTRODUCTION

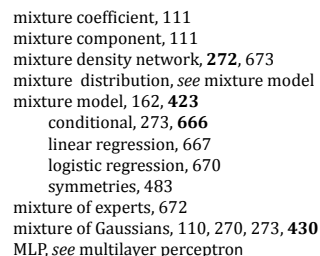
A back-of-the-book index is a collection of words or phrases, often alphabetically arranged to allow users to locate information in a given book. Creating an index for a non-fiction book is the responsibility of the author, but most authors do not actually do it. Usually indexing is done by freelancers hired by authors, publishers or an independent business which manages the production of a book, namely professional indexers. An index entry is an index term with its locators as shown in Figure 1. The locators are usually depicted by single page numbers, ranges of page numbers or even the section numbers, where a bold number indicates a more important reference. Besides, a good back-of-the-book index may also contain semantic relationships among those index terms. Typical relationships include 1) subcategory represented by subheadings such as those under “mixture

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.

Copyright 2013 ACM 978-1-4503-2263-8/13/10 ...\$15.00.

<http://dx.doi.org/10.1145/2505515.2505627>.



```
mixture coefficient, 111
mixture component, 111
mixture density network, 272, 673
mixture distribution, see mixture model
mixture model, 162, 423
    conditional, 273, 666
    linear regression, 667
    logistic regression, 670
    symmetries, 483
mixture of experts, 672
mixture of Gaussians, 110, 270, 273, 430
MLP, see multilayer perceptron
```

Figure 1: An back-of-the-book index snippet

model”; 2) homographs or synonyms indicated by “see”, e.g. “MLP, see multiple perceptron”; and 3) cross references indicated by “see” and “see also”. A good index is more than just an alphabetical list of all proper nouns (which is properly called a concordance) and their locations; it is a way of re-organizing the knowledge in a book that can guide a reader to the locations they most need to consult. So a good book index should be regarded as a creative work of a professional indexer, which reflects his/her understanding and analyzing of the book content; thus the indexing industry believe the book index creation cannot be replaced by computers¹.

Indexing softwares can help to sort and manipulate index; establish subheading sequences; restyle and amend entries; and keep track of what has been indexed where; but cannot really create indexes “automatically”, or even provide some facility to do content analysis and understanding. Recent development of keyword extraction should render new assistance to the book indexing. Surprisingly, very little attention has been drawn to book indexing by taking advantage of keyword extraction techniques except [4], which explored the automatic back-of-the-book index generation problem as a keyword extraction problem using supervised learning. However, a set of keywords is still far away from the book index. What really matters for the index is a term with its proper locators that point to the text. For example, in the book *Pattern Recognition and Machine Learning* [1], the term “hidden Markov model” appears in more than 20 pages while the actual index entry has only 2 pages as its locators. Keyword extraction only gives a list of keywords in a context-oblivious way, without pointing out from what context the keyword should be indexed. Back-of-the-book index generation needs to identify the exact contexts/pages of a term to ensure it’s informative or valuable enough for indexing. The new challenge is how to identify the proper locators given a keyword.

¹<http://www.backwordindexing.com/Comp.html>

Besides, whether the supervised learning approach is a good choice seems questionable. First, books from different domains vary a lot in vocabulary composition and structure style, requiring various indexing specialties. For example, there are different indexing guiding books for medicine [24] and law [9]. Second, book indexing is a highly subjective work and indexes of different books are always created by different professional indexers who have their own preferences and background [5]. Third, the learning is on an extremely unbalanced training set. As we found in our dataset, the index size is only 0.42% of the length of book in average. These deficiencies might cause the problems of domain dependence and harm the efficiency. All these give us the motivation to move a further step on book indexing by explore generating index terms aware of their locators through an efficient training-free and domain independent approach.

Admittedly, the intellectual and analytical work of indexing is the task of human, but can we do more than just give a list of keywords? Inspired by how human indexers work on book indexing, we presents a domain-independent and training-free approach, relying on the book content and structure and the web knowledge base. Given a book, the index terms will be selected according to 1) the indexability score measuring a word’s quality of being an index term for its context, combined with, 2) the context weight that measures the importance of a term’s context based on the book structure information. We redefine the back-of-the-book generation as an optimization problem aiming to find a set of index terms with good “indexability” in contexts with high context weight. We then develop efficient algorithms to solve the problem and evaluated our methods on hundreds of books in more than 10 domains, including computer science, art, history, Psychology, and so on. The experimental results show that our approach is effective and efficient in creating book indexes and outperforms the previous supervised learning approaches. Besides the domain-independent and training-free, we argue our approach could be easily applied to books of other languages.

2. RELATED WORK

2.1 Existing aiding tools

There are plenty of indexing softwares such as CINDE², Macrex², and IndDoc [14], which provide aid to human indexers. They can help to produce a concordance or a word list, locate the various occurrences of a word, sort the entries in alphabetical order and format the resulting index according to different index styles. However, they by no means give any aid to the key indexing process, i.e. selecting an index term based on understanding of book content.

2.2 Back-of-the-book index generation

Though more and more books are becoming available online in electronic format, the automatic back-of-the-book index generation, as an influential application for the indexing society and publishing industry and also a challenging AI task, has not receive enough attention. In 1998, Schutze proposed hypertext concordance as a hypertext back-of-the-book index, which mainly focus on the design of connecting index entries and their occurrence in a book while simply select index terms based on the likelihood ratio of occurring

²<http://www.asindexing.org/i4a/pages/index.cfm?pageid=3319>

in the target document collection and a reference collection [16]. Csomai and Mihalcea first evaluated the performance of different informativeness measurements such as TFIDF, χ^2 , and KL divergence for selecting book index terms [3], and then studied it in a supervised learning framework [4] based on syntactic features, linguistic features, encyclopedic features and so on. However, their work is closer to keyword extraction in long documents rather than build a real book index.

2.3 Keywords Extraction

The closest work of rich literature to back-of-the-book indexing is the keyphrase extraction, which has been extensively studied in different application contexts such as academical and digital library resources [6, 18, 8, 13], web pages [12, 25, 15, 19] and social media documents [11, 26, 20]. Presumably many of the keyphrase extraction methods can be applied to select candidate index terms for books. However, they are inherently context-oblivious, limiting its capability to find the appropriate references of index terms. What’s more, most of the keyphrase extraction approaches are either highly supervised or must be run in a large corpus environment. Our is not only unsupervised and domain independent, but also corpus-free, thus can be much more practical for back-of-the-book index generation.

3. METHODS

Unlike the previous approach that simply hires a brunch of features and then throws them to supervised learning models, ours is inspired by the way professional indexers create real book indexes. They read the text, select the terms that readers are likely to inquiry and relates them to the text segments, based on not only the structure and content of a book but also, more importantly, their own background knowledge that enables them to gain a comprehensive understanding of the book. Similarly, our method goes through the whole book sequentially and identifies the index terms according to the indexability score and context weight measured by both inner information within the book and external information from web knowledge.

3.1 Problem Statement

Definition A *book* is a sequence of contexts, i.e. $B = [C_0, \dots, C_{|B|-1}]$. A *context* is a syntactic unit of discourse such as a paragraph or a sentence, composed of a sequence of terms, denoted by $C_i = [t_{i1}, \dots, t_{i|C_i|}]$, where a term t is represented by an n-gram. The whole set of terms in book B is denoted by T . An index item is paired by an index term and its context, denoted by $it = (t, id_C)$, where $t = it[0] \in T$ and $id_C = it[1] \in \{0, \dots, |B| - 1\}$. A back-of-the-book index, denoted by $E = \{it_0, \dots, it_{s-1}\}$, is a collection of index items where s is the index size or total number of index items.

Given a book and its index size s , our goal is to generate s index items that maximize the indexability scores as well as have good coverage to all the information conveyed in the whole book. Note that the total number of index terms $\leq s$ since an index entry can have multiple contexts. Based on the definitions, the back-of-the-book index generation can be formulated as the following problem:

$$E^* = \arg \max_{E=\{it_0, \dots, it_{s-1}\}} \sum_{j=0}^{s-1} \omega(C_{it_j[1]}, B) \cdot S(it_j[0], C_{it_j[1]})$$

s.t.

$$\omega(C_{it_j[1]}, B) \geq \omega_0 \quad (1)$$

$$S(it_j[0], C_{it_j[1]}) \geq S_0 \quad (2)$$

$$\max_{0 \leq j \leq s} (|it_j[1] - it_{j-1}[1]|) \leq \sigma_0 \quad (3)$$

where

- $C_{it_j[1]}$ is the context of term $it_j[0]$, $it_j[1] \leq it_{j-1}[1]$ ($0 < j < s$, $it_{-1}[1] = 0$, $it_s[1] = |B| - 1$);
- $\omega(C_{it_j[1]}, B)$ measures the context weight score of $C_{it_j[1]}$ in book B ;
- $S(it_j[0], C_{it_j[1]})$ measures the indexability score of the term $it_j[0]$ in its context $C_{it_j[1]}$;
- ω_0 and S_0 are two threshold parameters and σ_0 is an adjustable parameter to control the coverage of book index.

Since our method is corpus-free, relying only on a given book, we can omit the B in all related notations.

3.2 Measuring Indexability

The indexability is encoded by two factors, context-aware informativeness and keyphraseness.

3.2.1 Context-aware Informativeness

Most known approaches to measure term informativeness fall into two categories: frequency-based and semantic-based. The frequency-based methods are based on derivations from term frequency (TF) and document frequency (DF), while the semantic-based methods compute latent semantic analysis (LSA) term vector length or the ratio of a term’s LSA vector length to its document frequency [10]. Those methods, all corpus-based, which are effective to identify informative words in document level, lose the power to capture the term informativeness in a particular context, since they are context-oblivious[22]. We could segment a book into chapters or paragraphs, and then apply the corpus-based methods, or even more sophisticated keyword extraction algorithms. However, we still will face the difficulty in determining proper locators of index terms. We could hire the context weighting metric, which will be given in next subsection, to choose the contexts. However, before leveraging the structure information of a book, can we derive a context-aware term informativeness measurement for a given context or even a general textual snippet? Our intuition is that higher informative terms in a context tend to have more similar contexts. Considering the following two contexts of “graph” from chapter 8 of PRML, it is easy to demonstrate that the first one tends to have more similar contexts on the web by search “graph” using Wikipedia, Google, and Bing. This motivates our definition of context-aware term informativeness.

1. A graph comprises nodes (also called vertices) connected by links (also known as edges or arcs).
2. We shall now go from this graph to the corresponding representation of the joint probability distribution written in terms of the product of a set of conditional distribution, one for each node in the graph.

Definition Given a term t and its context C , suppose the featured context set of t is $U_f = \{C_i | i = 1, \dots, k\}$, the term informativeness of t in context C_i is defined as

$$I(t, C) = \sum_i sim(C, C_i) \cdot p(C_i) \quad (4)$$

where $sim(C, C_i)$ is a similarity distance between C and C_i , and $p(C_i)$ is the prior importance of C_i to t .

Wikipedia is used as the knowledge base since it is currently one of the largest and most readily available knowledge repositories. Given any keyword, the Wikipedia query API will return the ranked Wikipedia entries along with the contexts containing the keyword. The top k results are considered as the featured context set and the discounted rank is used to estimate the prior $p(C_i)$.

3.2.2 Keyphraseness

We assume the keyphraseness of a term is qualitatively determined by two key properties: 1) it should be a meaningful phrase in the book; 2) it is highly possible that the term had been selected as a keyphrase in other similar resources. Pointwise Mutual Information (PMI) [2], was shown to be the best metric to measure word association [17]. Given an n -gram $(x_1..x_n)$, its PMI is defined as

$$PMI(x_1..x_n) = \log \frac{p(x_1..x_n)}{p(x_1)p(x_2..x_n)} \quad (5)$$

where $p(x)$ is the probability of occurrence of word x and $p(x_1..x_n)$ the corresponding joint probability in a book.

In Wikipedia a term is marked as “keyword” in terms of title or anchor text in a link. The Wikipedia keyphraseness estimates the probability of a term t being a keyword in a new document by the ratio of number of documents where t was marked as a keyword to its DF [4].

$$WK(t) = count(D_{key}(t))/DF(t) \quad (6)$$

This metric is computational impractical due to the high cost for harvesting $count(D_{key})$ online. It is even costly to precompute WK of all keywords in Wikipedia to provide local fast access. Thus we present a new lightweight version by leveraging the power of Wikipedia API.

$$WK(t) = count(Backlinks(t))/totalhits(t) \quad (7)$$

where $Backlinks(t)$ lists pages that link to t by Backlinks API³ and $totalhits(t)$ returns number of hit pages by Search API⁴.

3.3 Context weighting

It is possible that human indexers may not be knowledgeable enough to gain an exact understanding of every book they work on, especially the professional and technical books. However, they can still create an acceptable index using their indexing techniques, most of which are based on the fact the structure and format information in a book gives useful clues for what should be indexed. For example, words in chapter titles should be more likely to be indexed than those in general paragraphs. To leverage the structure information, the context weight score is defined to measure the importance of a context w.r.t indexability based on solely on structure or format features, which have been previously shown useful for keyphrase extraction [7]. We define the context weight as a weighed sum of the three indicators:

$$\omega(C) = \lambda_1 \cdot \omega_t(C) + \lambda_2 \cdot \omega_d(C) + \lambda_3 \cdot \omega_e(C) \quad (8)$$

where

³<http://www.mediawiki.org/wiki/API:Backlinks>

⁴<https://www.mediawiki.org/wiki/API:Search>

- ω_t is a boolean function determined by whether the context is a chapter or sub-chapter title;
- $\omega_d(C) = 1 - \frac{C_{id}(C) - C_{id}(title_C)}{N_{title_C}}$ measures the normalized distance from the context to its direct chapter or sub-chapter title, where $C_{id}(C)$ denotes the id of context C , $title_C$ the title of context C and N_{title_C} the number of contexts under $title_C$;
- $\omega_e(C) = \frac{|C_{cand}|}{|C|}$ measures the effective length of the context by counting the fraction of candidate index words where C_{cand} indicates all the candidate index words of C .
- λ_1 , λ_2 , and λ_3 are adjustable weights for the above three indicators.

3.4 Algorithm and Technical Implementation

Algorithm 1: Back-of-the-book index generation

```

1 Input: a book  $B$ 
2 Output: a book index  $E$  sizes  $s$ 
3 begin
4   initialize a temporary index  $\hat{E}$ ;
5   for  $C \in B$  do
6      $cands \leftarrow getCandidates(C)$ ;
7      $\omega_C \leftarrow getContextWeight(C, cands, B)$ ;
8     if  $\omega_C < \omega_0$  then continue;
9     for  $t \in cands$  do
10       $S \leftarrow getIndexability(t, cands, C)$ ;
11      if  $S < S_0$  then continue;
12       $Score \leftarrow \omega_C \cdot S$ ;
13      add  $((t, id_C) : Score)$  to  $\hat{E}$ ;
14   move top  $s$  highest Score items from  $\hat{E}$  to  $E$ ;
15   while  $\exists it_j$  s.t.  $it_j[1] - it_{j-1}[1] > \sigma_0$  in  $E$  do
16      $S_1 \leftarrow \{it \in \hat{E} | it_{j-1}[1] \leq it[1] \leq it_j[1]\}$ ;
17     move arg max  $Score(it)$  from  $\hat{E}$  to  $E$ ;
18      $S_2 \leftarrow \{it_k \in E | it_{k+1}[1] - it_{k-1}[1] \leq \sigma_0\}$ ;
19     remove arg min  $Score(it)$  from  $E$ ;

```

An efficient algorithm for the back-of-the-book index generation problem is presented in Algorithm 1. *getCandidates* returns the candidate index terms of a context based on POS patterns and stop-words. *getContextWeight* calculates the context weight score defined in Eq.(8). The default λ_1 , λ_2 , and λ_3 are set to 0.5, 0.3 and 0.2. ω_t , ω_d , and ω_e are all initialized to zero. The effective length is calculated by the ratio of number of candidates to the total number of grams in context C , shown in line 5. ω_t and ω_d will be set to 1 if the context is a chapter or subchapter title. All the chapter and subchapter titles of a book are collected by detecting its table of content [23]. We then simply check whether C can find a fuzzy match in the title collection. If C is not a title, we then set ω_t to 0 and compute ω_t based on the distance of C deviated from its direct title. We find the $title_C$ by searching the nearest context before C from the title collection. Number of contexts under a title is the difference between id of a title and id of its next sibling title.

getIndexabilityScore computes the indexability score of each candidate index term in its context. The default implementation of *Sim* for context-aware term informativeness

score is the cosine similarity using candidate index terms as the feature vector. The similarity score is discounted by $\log(rank + 1)$ as an approximation of the prior probability p in Eq. (4). Finally, we use a hybrid score combining the context-aware informativeness with term frequency, PMI, and Wikipedia Keyphraseness. In the experiments, we use a simple linear combination with the same weight and each part normalized to $[0,1]$. We use *TF* instead of *TFIDF* because we found that *TF* is comparable to *TFIDF* for identifying index terms. Moreover, it is costly to have a huge word dictionary with *DF* that covers all candidate index term for an arbitrary book while computing *TF* is super easy and fast. The *PMI* score is computed according to Eq. (5) while the probability of an n-gram is simply estimated using its *TF* in a book. The Wikipedia keyphraseness is based on Eq. (7).

The smaller ω_0 and S_0 are, the more items will be added to \hat{E} . Their default values are set to 0.01. The final s items are tweaked to satisfy Eq. (3), i.e., making the distance of each two adjacent items no larger than ω_0 , delineated by line 15-19. In line 16-17, we select from \hat{E} the item of the highest Score between the two items whose distance is large than ω_0 and add it to E . In line 18-19, we remove from E the item of the lowest Score while the distance between its previous and next item is no larger than ω_0 . We set ω_0 by the minimum number of contexts in a subchapter.

4. EMPIRICAL EVALUATION

4.1 Datasets and Experimental Setup

To evaluate our back-of-the-book index generation method, we conduct extensive experiments on books in various domains, from the Gutenberg dataset and the open book dataset described in Table 1. The first one was created by [4], containing 55 free books collected from Gutenberg. Every book in the dataset has the body text, the original version and three cleaned versions of its index. The cleaned versions are a list of keyphrases from the original index without their locators. However, there is no page number information in the body text. So on this dataset we cannot consider the location of index terms, but only serve the evaluation as a keyword extraction task. Books in the second dataset was collected from the Citeseer repository, most of which are in computer science and engineering [21]. We manually select 213 books with good quality back-of-the-book index. The original books are all in pdf format. We extracted the pagged body text and the back index using Pdfbox. Having each index term associated with its locators (page numbers), we can perform fairer evaluation for different methods, not solely in the keyword extraction style.

4.2 Candidate Index Terms Generation

The POS patterns can be represented by regular expression “[IJN]{0,4}N”. A single letter ‘N’ represents a word whose tag starting with ‘N’, belonging to a Noun. ‘J’ indicates an Adjective. ‘I’ denotes a Preposition. “[IJN]{0,4}N” contains all the n-grams ($n \in [1, 5]$) where the last word is a Noun and all the previous ones are Adjective, Preposition or Noun. When tagging, we use the model ‘wsj-0-18-left3words-distsim.tagger’ from the Stanford POS tagger⁵. We first examine whether our candidate index term gen-

⁵<http://nlp.stanford.edu/software/tagger.shtml>

Table 1: Datasets for book index generation evaluation

Dataset	#Books	#Words	#Contexts	Main domains
Gutenberg	55	7,164,463	301,581	History, Art, Psychology, Philosophy, Literature, Zoology
Open Book	213	22,279,530	1,135,919	Computer Science, Engineering, Information Science

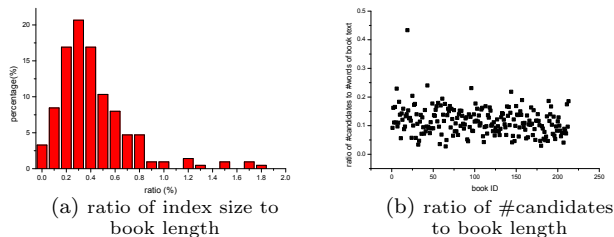


Figure 2: Results for candidate index term generation

eration method is capable to enumerate all potential index terms. Ideally, we hope it can cover 100% of the true index terms by generating a relative small number of candidates, since the number of items from a book index is much smaller than the number of words from the book text. We estimated the average ratio as 0.42% in our open book dataset, which is very close to 0.45% shown in the previous work[4]. The distribution of the ratio is shown in Figure 2a. Figure 2b shows the ratio of number of generated candidates to the total number of words in each book. It covers more than 95% of the ground truth using only less than 25% of number of all the words in most books. In average, there are 104600 words in a book while number of candidates is 12055, in a ratio of 11.5%. Considering that those candidates are selected from unigram to five-gram, it actually rules out far more than 88.5% of the total grams.

4.3 Index Generation

To measure the effectiveness of our approach as well as the roles of “indexability” and “context weight”, we run experiments on the open book dataset using three methods, showing in Figure 3a. CTI is the context-aware term informativeness in Section 3.2.1. Indexability does not use context weight. CI-Indexer represents method shown in Algorithm 1. The results clearly demonstrate that *TF*, *PMI*, and *WK* can gain around 4% more recall over CTI while context weight can bring around 6% more recall over indexability. Note that here the recall of CIT is lower than that in Section 4.3 because we take the locators of index terms into consideration. Only a index term with its context in the right page will be counted into recall.

We then compare our method with three baselines including TFIDF, KEA⁶, and SLD (supervised learning using decision tree in Csomai’s [4]). KEA builds a Naive Bayes model using features TFIDF, first occurrence, length of a phrase, and node degree (number of candidates that are semantically related to this phrase). First occurrence is computed as the percentage of the document preceding the first occurrence of the term in the document. We compute the node degree as the textrank[13] degree in a book by simply relating two candidate terms with each other if they are in the same context. For SLD, we hire all the features presented in [4] except the discourse comprehension based features which are too complicate to implement. We choose decision tree because its training is much faster than other two models in [4] while its performance is close to the best. Since those three are context-oblivious, we set two strategies to make them context-aware. First, we select the page of a term’s

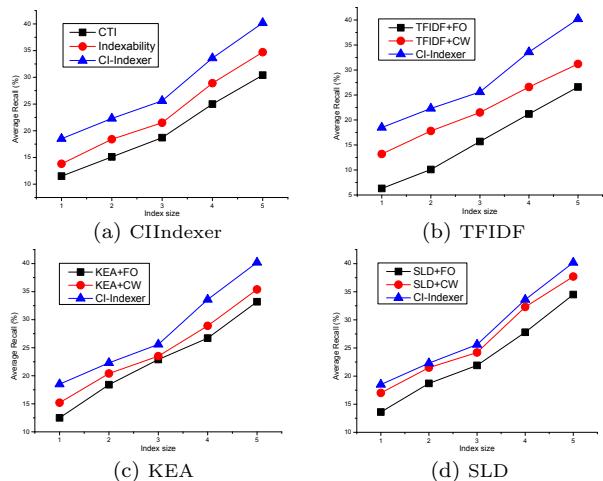


Figure 3: Results for book index generation

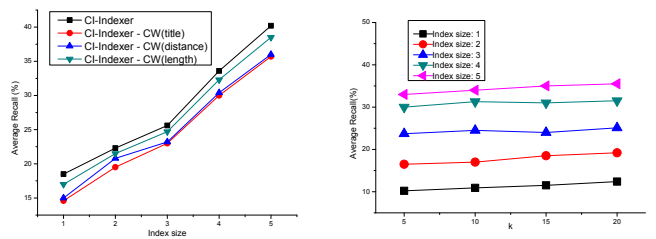
first occurrence as its locator, denoted by “+FirstOccurrence” in Figure 3. Second, we apply our context weight to them, denoted by “+CW”. The results are shown in Figure 3b, 3c and 3d respectively. For all the three baselines, adding context weight gains better performance than using the simple first occurrence guess, especially for TFIDF. KEA benefits least from context weight, suggesting its first occurrence and node degree features play part of similar role as our context weight features. SLD outperforms TFIDF and KEA under both the two strategies probably because the new features of POS pattern and Wikipedia keyphraseness. SLD+CW performs closest to ours.

4.4 Parameters and Running Time

The context weight can be regarded as an “indexability” measurement in context level. To further understand the importance of different ingredients in context weight, we run the CI-Indexer by ruling out one part of context weight and set the other two equally weighted. The experimental results are shown in Figure 4a denoted by CI-Indexer - CW(title), CI-Indexer - CW(distance) and CI-Indexer - CW(length). Based on the decrement from CI-Indexer to each of them, we could conclude that “title” is more important than “distance” while “length” is the least important part. We also study the sensitivity of their weights λ_1 , λ_2 and λ_3 and recommend their setting following $\lambda_1 > \lambda_2 > \lambda_3$ with $\lambda_1 \in [0.4, 0.6]$, $\lambda_2 \in [0.2, 0.4]$ and $\lambda_3 = 1 - \lambda_1 - \lambda_2$. Another important parameter is the number of retrieved snippets from Wikipedia when computing the context-aware term informativeness. We run the CTI in Section 4.3 by setting the number to 5, 10, 15 and 20. Results in Figure 4b shows the performance of CTI is not sensitive to this number but gain slight improvement as it increases under different output index size.

Time is not an issue for our approach since it is training free and corpus free. For a book of around 200,000 words, while our approach totally costs 5-8 minutes to generate an index sizing 5 times of the ground truth, the supervised learning approach costs 10-15 minutes apart from the much more offline training time. More than 60% time of our approach is cost on the computation of CTI. In terms of scala-

⁶<http://www.nzdl.org/Kea/>



(a) Context weights (b) number of retrieved snippets
Figure 4: Parameters Study

bility, the proposed method is inherently parallelizable, not only at the document level, but also at the context level, since computing CTI does not depend on any other context in the document. In addition, we do not need to issue the same query more than once. Our strategy is to locally cache the returned results of every seen query. For a new term seen in a previous query, we can directly access the local cached file. If we have built a large local pool, the queries will rarely go to a search engine or other source. Given a corpus size N (words in total), the number of actual issued queries will be at most the number of unique terms, which is far less than $O(N)$. Of course, new terms never seen will have to be processed, but there will be fewer of these over time.

5. CONCLUSION AND FUTURE WORK

In this paper, we tackled the back-of-the-book index generation problem in a novel context-aware approach, which is capable of giving index terms as well as their locators. We measure *indexability* of a term in context by *informativeness* and *keyphraseness*. We then developed a new context-aware term informativeness measurement by leveraging web knowledge base and a computationally practical Wikipedia keyphraseness. The structure information residing in book text are then encoded as context weight to weight the importance of contexts. We conducted extensive experiments to examine the performance of our approach.

In the future, we will make our back-of-the-book index generation system publicly available. Besides, the automatic construction of semantic relationships between index terms, which are not addressed in this work, including subheadings, homographs or synonyms, and cross references, could all be promising future research topics.

6. REFERENCES

- [1] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [2] K. W. Church and P. Hanks. Word association norms, mutual information and lexicography. *Computational Linguistics*, 16(1):22–29, 1990.
- [3] A. Csomai and R. Mihalcea. Investigations in unsupervised back-of-the-book indexing. In *Proceedings of FAIRS*, pages 211–216, 2007.
- [4] A. Csomai and R. Mihalcea. Linguistically motivated features for enhanced back-of-the-book indexing. In *Proceedings of ACL*, pages 932–940, 2008.
- [5] V. Diodato and G. Gandt. Back of book indexes and the characteristics of author and nonauthor indexing: Report of an exploratory study. *JASIS*, 42(5):341–350, 1991.
- [6] I. H. W. E. Frank, G. W. Paynter and C. Gutwin. Domain specific keyphrase extraction. In *Proceedings of IJCAI*, pages 668–673, 1999.
- [7] K. Hofmann, M. Tsagkias, E. Meij, and M. de Rijke. The impact of document structure on keyphrase extraction. In *Proceedings of CIKM*, pages 1725–1728, 2009.
- [8] E. F. C. G. I. H. Witten, G. W. Paynter and C. G. Nevill-Manning. Kea: practical automatic keyphrase extraction. In *Proceedings of the fourth ACM conference on Digital libraries*, pages 254–255, 1999.
- [9] P. Kendrick and E. L. Zafran. *Indexing Specialities: Law*. Medford, NJ : Information Today, Inc., 2001.
- [10] K. Kireyev. Semantic-based estimation of term informativeness. In *Proceedings of NAACL*, pages 530–538, 2009.
- [11] Z. Li, D. Zhou, Y.-F. Juan, and J. Han. Keyword extraction for social snippets. In *Proceedings of WWW*, pages 1143–1144, 2010.
- [12] R. Mihalcea and A. Csomai. Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of CIKM*, pages 233–242, 2007.
- [13] R. Mihalcea and P. Tarau. Textrank: Bringing order into texts. In *Proceedings of EMNLP*, pages 404–411, 2004.
- [14] A. Nazarenko and T. A. E. Mekki. Building back-of-the-book indexes. *Terminology*, 11(1):199–224, 2005.
- [15] E. F. O. Medelyan and I. H. Witten. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of EMNLP*, pages 1318–1327, 2009.
- [16] H. Schutze. The hypertext concordance: a better back-of-the-book index. In *Proceedings of Computerm*, pages 101–104, 1998.
- [17] E. Terra and C. L. Clarke. Frequency estimates for statistical word similarity measures. In *Proceedings of NAACL*, pages 165–172, 2003.
- [18] P. D. Turney. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2(4):303–336, 2000.
- [19] R. West, D. Precup, and J. Pineau. Automatically suggesting topics for augmenting text documents. In *Proceedings of CIKM*, pages 929–938.
- [20] W. Wu, B. Zhang, and M. Ostendorf. Automatic generation of personalized annotation tags for twitter users. In *Proceedings of NAACL*, pages 689–692, 2010.
- [21] Z. Wu, S. Das, Z. Li, P. Mitra, and C. L. Giles. Searching online book documents and analyzing book citations. In *Proceedings of DocEng*, 2013.
- [22] Z. Wu and C. L. Giles. Measuring term informativeness in context. In *Proceedings of NAACL*, pages 259–269, 2013.
- [23] Z. Wu, P. Mitra, and C. L. Giles. Table of contents recognition and extraction for heterogeneous book documents. In *Proceedings of ICDAR*, 2013.
- [24] L. P. Wyman. *Indexing Specialities: Medicine*. Medford, NJ : Information Today, Inc., 1999.
- [25] Y. Z. Z. Liu, W. Huang and M. Sun. Automatic keyphrase extraction via topic decomposition. In *Proceedings of EMNLP*, pages 366–376, 2010.
- [26] W. X. Zhao, J. Jiang, J. He, Y. Song, P. Achananuparp, E.-P. Lim, and X. Li. Topical keyphrase extraction from twitter. In *Proceedings of ACL*, pages 379–388, 2011.