Unsupervised Analysis of Web Page Semantic Structures by Hierarchical Bayesian Modeling

Minoru Yoshida[1], Kazuyuki Matsumoto[1], Kenji Kita[1], and Hiroshi Nakagawa[2]

[1] Institute of Technology and Science, University of Tokushima
2-1, Minami-josanjima, Tokushima, 770-8506, Japan
{mino,matumoto,kita}@is.tokushima-u.ac.jp
[2] Information Technology Center, University of Tokyo
7-3-1, Hongo, Bunkyo-ku, Tokyo 113-0033, Japan
nakagawa@dl.itc.u-tokyo.ac.jp

Abstract. We propose a Bayesian probabilistic modeling of the semantic structures of HTML documents. We assume that HTML documents have logically hierarchical structures and model them as links between blocks. These links or dependency structures are estimated by sampling methods. We use hierarchical Bayesian modeling where each block is given labels such as “heading” or “contents”, and words and layout features (i.e., symbols and HTML tags) are generated simultaneously, based on these labels.

Keywords: Hierarchical Bayesian modeling, Web document analysis, Gibbs sampling.

1 Introduction

In this study, we propose a new model for HTML documents that can extract *document structures* from them. *Document structures* are hierarchical structures of documents that decompose documents into smaller parts recursively. For example, scientific papers typically consist of several *sections*, each of which can be decomposed into *subsections*. In addition, *titles*, *abstracts*, and so on, are included in the document structure.

Web document analysis is a challenge to extract such document structures from *HTML documents*. Web documents can be decomposed into *subdocuments*, typically with their *headings* representing titles of each subdocuments. Figure 1 shows an example of subdocuments found in the web page shown in Figure 2, where each subdocument is presented as a heading. For example, in this document, “Age: 25” is a subdocument with the heading “Age” and content “25”. Our purpose is to extract such lists of subdocuments such that those in the same list have parallel relations (such as “TEL:...” and “FAX:...” in Figure 1.) Note that there are “nested” lists – the element starting with “Contact:” contains another list “TEL:...” and “FAX:...”.

We assume generative models for documents. The most basic way to collect parallel subdocuments is to use clustering algorithms such as K-means.

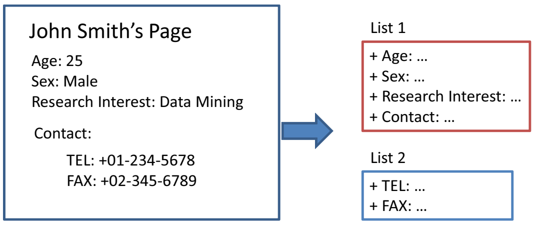

Fig. 1. Example Document and Its Repeated Tuples

The model we propose in this study is somewhat similar; however, it uses a hierarchical Bayesian framework to not only model similarities of visual effects, but also model

- hierarchical structures by using dependency trees as constraints to prior probabilities, and,
- simultaneously model local tag similarity and general visual effect usage, by using the hierarchical Bayesian model.

2 Related Work

Research on extracting repeated patterns from Web documents has a long history. The most popular approach is to make DOM trees and find frequency patterns on them[1, 2]. The problem with this approach is that it does not work for repeated patterns indicated by non-DOM patterns including patterns with symbols as shown in Figure 2.

Several studies have addressed the problem of extracting logical structures from general HTML pages *without labeled training examples*. One of these studies used domain-specific knowledge to extract information used to organize logical structures [3]. However, the approach in these studies cannot be applied to domains without any knowledge. Another study employed algorithms to detect *repeated patterns* in a list of HTML tags and texts [4, 5] or more structured forms [6–8] such as DOM trees. This approach might be useful for certain types of Web documents, particularly those with highly regular formats such as `www.yahoo.com` and `www.amazon.com`. However, there are also many cases in which HTML tag usage does not have significant regularity, or the HTML tag patterns do not reflect semantic structures (whereas symbol patterns do.) Therefore, this type of algorithm may be inadequate for the task of heading extraction from arbitrary Web documents. Nguyen [9] proposed a method for web document analysis using supervised machine learning. However, our proposal is to use probabilistic modeling for Web documents to obtain their structures in an unsupervised manner.

Some studies on extracting titles or headlines have been reported in [10, 11]. Our task differs from these, in that their methods focus only on titles

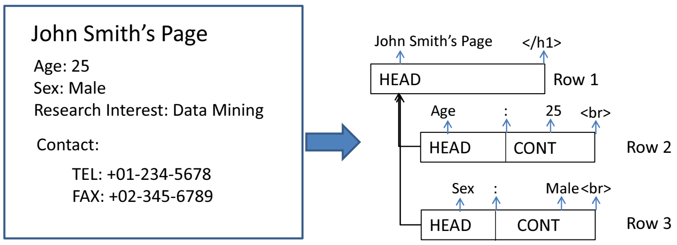

Fig. 2. Conversion from HTML Document to Dependency Structure. HEAD represents "heading" and CONT represents "contents".

(and headlines) and ignore the other parts of Web documents, while our algorithm handles *all parts* of the Web documents and provides a tree structure of the entire document; this algorithm enables the system to extract various types of heading other than titles and headlines, such as attributes. In particular, our approach has the advantage that it can handle symbols as well as HTML tags, making the system applicable to many private (less formal) Web documents.

Cai et al. [12] proposed VIPS that can extract content structure without DOM trees. Their algorithm depends on several heuristic rules for visual representation of each block in Web documents. However, we propose unsupervised analysis based on a Bayesian probabilistic model for Web document structures. This has several advantages, including easy adaptation to some specific document layouts, easiness of tuning its parameters (because we only have to change hyperparameters), and ability of obtaining probabilities for words and symbols that may be used for other type of documents such as text files.

Weninger et al.[13] compared several list extraction algorithm. One of the contributions of this study is that we propose a model for *nested structures* of lists, which has not been tried in most previous studies.

3 Preliminaries

3.1 Problem Setting and Terms Definition

We model an HTML document as a list of *rows*. Each row r_i is a list of *blocks*, i.e, $r_i =< b_{i_1}, ..., b_{i_{|r_i|}} >$. Here $|r_i|$ is the *size* of row r_i, which is the number of blocks in the row. A block b_j is a pair $(w_j, \mathbf{s}_j)$ of the *representative word* and *symbol list* $\mathbf{s} =< s_{j_1}, ..., s_{j_{|\mathbf{s}_j|}} >$.

Because our experiments currently use Japanese documents, which do not contain word breaking symbols, it is not a trivial task to extract the representative word for each document. In our current system, we extract the predefined length of suffix from each block[1] and call them *representative words*. We did

[1] Length changes according to character types such as "trigrams for alphabets and numbers", "unigrams for Kanji characters", etc.

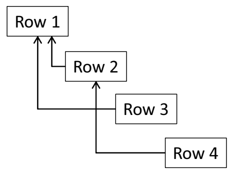

Fig. 3. Prohibited Dependency Relations

not use word breaker tools, partly because they are not for short strings such as those that frequently appear in Web documents, partly because we do not need human-comprehensive features (because our purpose is not information extraction but document structure recognition), and partly because simple suffix extraction rules contribute to stability of extraction results.

We assume several hidden variables associated with the above observed variables. First, each block b_j is labeled with *label* l_j, which can have one of two values $\{H, C\}$. Here, H means *heading* and C means *contents*. Headings are titles of subdocuments, and we assume that the heading is presented in the first few blocks of the subdocument, followed by other blocks that we call content blocks. (See Figure 2.) Blocks labeled with H are called *heading blocks*, and blocks labeled with C are called *content blocks*. *Heading rows* are the rows that contain one or more heading blocks, while *content rows* are the rows that contain no heading blocks. In addition a hidden variable p_k is associated with each symbol s_k. It indicates whether the symbol is a linefeed-related one, used in the Gibbs sampling step described later.

Next, we assume the dependency structures in documents. Here a *dependency relation* between two rows means that one row is modified by another. In Figure 2, the row “Age: 25” (*depending row*) depends on the row “John Smith” (*depended row*). We assume that a pair of hidden variables $(dep_i, bound_i)$ for the i-th row. (We also write $dep(r_i) = j$ if $dep_i = j$.) Here, dep_i is the row id that the i-th row depends on. Note that the structure is augmented with an additional variable $bound_i$, which denotes the position of the *boundary* between heading blocks and content blocks in the i-th row. If $bound_i = 0$, it means that there is no heading block in the row (and $dep_i = -1$ in this case), and if $bound_i = |r_i|$, it means that there is no content block in the line. The *sisters* of row r_i denote the list of all the rows r_{i1}, r_{i2}, ... depending on the same row as r_i (i.e., $dep(r_{i1}) = dep(r_{i2}) = ... = dep(i)$).

Dependency Structures. Our definition of *dependency structures* in this study is slightly different from that used in natural language processing communities. One main difference is that it allows *isolated rows* that do not depend on any other rows. We consider that isolated rows link to a special row called the *null row*, indicated by the id number -1. We consider that two dependency structures are different if at least one row has different links. Note that we prohibit crossings in the dependency structures, and their probability is set to 0 (See Figure 3).

```
[1] R_nolink -> CL (1.0)
[2] [2.1] R_haslink -> HL CL (p4) | [2.2] HL (p5)
[3] [3.1] HL -> H HL (p6) | [3.2] H (p7)
[4] [4.1] CL -> C CL (0.5) | [4.2] C (0.5)
[5] H -> word1 | word2 | ...
[6] C -> word1 | word2 | ...
```

Fig. 4. PCFG Rules

Dependency structures with no crossing links are called *possible dependency structures.*

4 Probability Model and Estimation

We define the probability of generating the whole document as $P(d, T) = P_{prior}(T) \cdot P_{block}(d|T)$, where T is the assignment of $(dep, bound)$ for all rows in document d.

Our idea is to divide the process of generating blocks into *vertical* and *horizontal generation* processes. The former generates rows under the constraints of row dependency structures. Currently, the probability of row dependency structures is defined as uniform distribution among all possible dependency structures. After each row is generated, all blocks in the row are generated horizontally with probabilities induced by CFG rules. Dividing the generation process in this way reduces the size of the search space. One of the merits of using CFG for prior probability calculation is that it can naturally model the ratio of headings and contents in each row, regardless of how many blocks are in the line. For example, if we directly model the probability of the value of $bound_i$, different lengths of rows require different models, which makes the model complicated. Instead, in our model, the ratio can be modeled by generation probabilities of a few of rules.

Figure 4 shows our PCFG rules used in our model. `H` means *headings* and `C` means *contents*. `HL` and `CL` mean *heading list* and *contents list*, which generates a list of heading blocks and content blocks, respectively. `R` means *rows*, which consist of one heading list, optionally followed by a content list. Here `R_nolink` is a nonterminal that indicates content (isolated) rows, and `R_haslink` is a nonterminal for heading rows. Note that this model prohibits headings from starting in the middle of each row.

Then, probabilities are calculated on the basis of the resulting CFG tree structures, using the PCFG rules shown in Figure 4.

Probability for Heading Rows. A heading row needs a probability of `p4` or `p5` before generating its heading and content lists. However, content rows do not need such probability (because they generate content lists with a probability of 1 by rule `[1]`.)

Probability by Heading Blocks. As rule [3] shows, for each heading row, the last heading block needs a probability of p7, and other heading blocks need a probability of p6, to be generated. We define *heading probability* of the row as $P_h(r) = p6^{n_h(r)-1} \cdot p7$, where $n_h(r)$ is the number of heading blocks in row r.

Probability by Content Blocks. From rule [4], it is easily shown that each content block needs a probability of 0.5 to be generated. We define *content probability* of the row as $P_c = 0.5^{n_c}$, where n_c is the number of content blocks in the document.

4.1 Block Probability

The remaining part of the probability is the probability for blocks $P_{block}(D|T)$. First, note that each block is labeled H or C, according to the CFG tree in the horizontal generation. Each word in the block is generated from a distribution selected according to this label. We assign one of the labels $\{B, N, E, L\}$ to each symbol in the document using the following rules. Intuitively, B denotes *boundary* between heading and contents, N denotes *non-boundary*, E denotes *end of subdocuments*, and L denotes *line symbols* that are used in most of the linefeeds, which are not likely to have any semantic meaning, such as heading-associated tags such as <h1>.

– If the separator s_{i_k} is in the last block of row r_i, and the value of p_{i_k} is 1, then it is labeled L.
– If the separator s_{i_k} is in the last block of row r_i, the value of p_{i_k} is 0, and $bound_i \neq i_k$, then it is labeled E.
– Otherwise, if $bound_i = i_k$, separators in block b_{i_k} are labeled B.
– Otherwise, separators are labeled N.

Based on these labels, $P_{block}(d|T)$ is defined as a multinomial distribution of the bag of words: $P_{block}(d, T) = P(\mathbf{w}_H) \cdot P(\mathbf{w}_C) \cdot P(\mathbf{s}_B) \cdot P(\mathbf{s}_N) \cdot P(\mathbf{s}_E) \cdot P(\mathbf{s}_L)$ where $\mathbf{w}_H$ is a list of words labeled H, $\mathbf{w}_C$ is a list of words labeled C, $\mathbf{s}_B$ is a list of symbols labeled B, $\mathbf{s}_N$ is a list of symbols labeled N, $\mathbf{s}_L$ is a list of symbols labeled E, $\mathbf{s}_L$ is a list of symbols labeled L, in document d.

Our word/symbol generation model is a hierarchical Bayesian model. We assume the following generative process for words assigned nonterminal H.

1. Each word w in a document labeled H (i.e, $w \in \mathbf{w}_H$) is drawn from the distribution H_d: $w \sim H_d$.
2. H_d, the heading distribution for document d, is drawn from the Dirichlet distribution with base distribution H_{base} and concentration parameter α_H: $H_d \sim Dir(\alpha_H H_{base})$.
3. The base distribution H_{base} is drawn from the Dirichlet distribution with measure B: $H_{base} \sim Dir(B)$.

Words labeled C, and separators labeled N, E, or L are distributed in the same manner. Base distributions and concentration parameters for them are denoted by C_{base}, N_{base}, E_{base}, and L_{base}, and α_C, α_N, α_E, α_L, respectively.

Sampling from B is slightly different, because distributions are generated not for each document, but for each *sister*. Here sisters are a group of rows that depends on the same row.

1. Each separator s in the class labeled B (i.e., $s \in \mathbf{s}_B$) is drawn from B_i: $s \sim B_i$.
2. B_i, the bound distribution for sisters class i, is drawn from the Dirichlet distribution with base distribution B_{base} and concentration parameter α_B: $B_i \sim Dir(\alpha_B B_{base})$.
3. The base distribution B_{base} is drawn from the Dirichlet distribution with measure B_B: $B_{base} \sim Dir(B_B)$.

Parallel subdocuments tend to have similar layouts, and such similarity typically appears in the boundary between headings and contents. We intend to model similar layouts by modeling boundary separators in the same list as that drawn from the same distribution.

We collapse the distribution for each document drawn from base distributions. For example, assume that $w_1, w_2, ..., and w_{n-1}$ have been drawn from H_d. Then, distribution for w_n is obtained by integrating out the multinomial distribution H_d, which results in the following equation.

$$P(w) = \frac{n_w}{\alpha_H + n_.} + \frac{\alpha}{\alpha_H + n_.} H_{base}(w) \tag{1}$$

where n_w is the number of times w occurs, and $n.$ is the number of all word occurrences, in the list $w_1, ..., w_{n-1}$. This equation can be obtained using the one for Pitman-Yor process [14] by assigning the discount parameter to be zero. By using this backoff-smoothing style equation, we can model the locality of the usage of words/separators by the first term, which corresponds to the number of occurrences of w in the same context, and global usage by the second term.

4.2 Sampling of Dependency Relations

Gibbs sampling is executed by looking at each row r_i and sampling the pair $(dep_i, bound_i)$ for the row according to the probability $P((dep_i, bound_i)|d_{-i})$. Here d_{-i} means the document without current row r_i. We calculate the relative probability of the document for all possible values for $(dep_i, bound_i)$ by taking all possible values for $(dep_i, bound_i)$, calculating the probability $P(d, T)$ for each dependency value, and normalizing them by the sum of all calculated values.

4.3 Sampling of Base Distributions

Another important part of our Gibbs sampling is sampling distributions given the document structures, which model the *general tendency of usage* of words and symbols. We use the fast sampling scheme described in [14] which omit time-consuming "bookkeeping" operations for sampling base distributions. First, the parameter $\mathbf{m}$, which indicates "how many times each word w was drawn *from the second term* of the equation 1, is drawn from the following distribution. $p(m_w = m|\mathbf{z}, \mathbf{m}^{-w}, \beta) = \frac{\gamma(\alpha\beta_w)}{\gamma(\alpha\beta_w + n_w)} s(n_w, m)(\alpha\beta_w)^m$ where $s(n,m)$ are unsigned Stirling numbers of the first kind. (Note that the factor k in Teh's representation corresponds to word w in our representation.) After drawing $\mathbf{m}$, the base parameter β is drawn from the following Dirichlet distribution: $(\beta_1, ..., \beta_K) \sim Dir(m_1 + \alpha'\gamma_1, ..., m_K + \alpha'\gamma_K)$ where α and γ are the strength parameter and the base distribution for the distribution for drawing H, respectively.

The following base distributions are sampled by using this scheme: H_{base} is sampled from $\mathbf{w}_H$, C_{base} is sampled from $\mathbf{w}_C$, N_{base} is sampled from $\mathbf{s}_N$, B_{base} is sampled from $\mathbf{s}_N$, and L_{base} is sampled from $\mathbf{s}_L$.

5 Implementation Issues

5.1 Sentence Row Finder

In HTML layout structure detection, *sentence blocks* are critical performance bottlenecks. For example, it is relatively easy to detect the suffixes of the rows that indicate sentences. However, it is difficult to decide whether the row *starts* with headings, especially when the sentences are decorated with HTML tags or symbols. (e.g., "Hobby: I like to hear music!")

Our idea is to use *prefixes* to decide whether the row contains headings. We assume that *rows starting with sentences contain no headers*, and the algorithm finds sentences by using the *ratio of obvious sentences* in all rows, starting with the prefix. The obvious sentences are detected by using simple heuristics that "if symbols in the row are only commas and periods, then the row surely consists of only sentences." Currently, if the ratio exceeds the threshold value 0.3, the row is determined as a sentence. Note that the sentence row finder is also applied to the baseline algorithm described later.

5.2 Hyperparameter Settings

We assume a Dirichlet prior for each rule of probability where Dirichlet hyperparameters are set $\alpha_1 = \alpha 2 = 5.0$ for rules `[2.1]` and `[2.2]`, and $(\alpha_1, \alpha_2) = (10.0, 90.0)$ for rules `[3.1]` and `[3.2]` heuristically. The latter parameters suggest that our observation that the length of heading lists is not so large, giving high-probability values to short heading lists. This sampling of parameters helps to stabilize sampled dependency relations.

5.3 Parallelization

Parallelization of the above Gibbs sampling is straightforward because each sampling of tuples $(dep, rel, bound)$ only uses the state of other tuples in the same document, along with the base distributions such as H_{base} and B_{base}, which are not changed in tuple sampling. The task of sampling tuples is therefore divided into several groups as each group consists of one or more whole documents, and the sampling of tuples for each group is executed in parallel. (Sampling base distributions is not easily parallelized.)

5.4 Dependency Structure Estimation

Gibbs sampling can be used as a scheme for *sampling* the latent variables; however, it is not obvious how to extract *highly probable* states using this sampling scheme. Plausible base distributions can be obtained by taking several samples and averaging them. However, dependency structures are so complicated that it is almost impossible to see the same sample of structure two times or more. We thus use the following heuristic steps to obtain highly probable structures.

– Run the Gibbs sampling for some burn-in period.
– Take several samples for base distributions, and average them as an estimation for the base distribution and PCFG rule probabilities (these parameters are fixed thereafter.)
– Initialize the latent categorical variables.
– Run the Gibbs sampling again, but only for categorical variables for some burn-in period and calculate the marginal likelihood of the selected structures in each time.
– Take the structures with the maximum marginal likelihood so far.
– Greedy finalization: for each line, fix the state to the one with the highest probability. This step is executed over all rows sequentially, and repeated several times.

6 Experiments

Our corpus consists of 1,012 personal web pages found in the Japanese web site `@nifty`. We randomly selected 50 Web documents from them. We excluded 10 documents that contain `<table>` tags because table structures need special treatment for proper analysis and including them into the corpus harms the reliability of the evaluation. We extracted all repeated subdocuments in the remaining 40 documents manually. Among them, 14 documents contained no repeated subdocuments. For each algorithm, we extracted each set of sisters from dependency structures and regarded them as resulting sets of lists. We used purity, inverse purity, and their f-measure for evaluation, which is a popular measure for clustering evaluation.

6.1 Evaluation Measure

To evaluate the quality of extracted lists, we use purity and inverse purity measures[15], which are popular for cluster evaluation. We regard each extracted list as a cluster of subdocuments and represent it with the pair (i, j), where i is the start position and the j is the end position of the subdocuments.

The end position is set just before the start position of the next subdocument in the list.[2] Subdocument extraction is evaluated by comparing this cluster to manually constructed subdocument clusters.

Assume that C_i is a cluster in the algorithm results, and L_i is a cluster in the manual annotation. Purity is computed by $P = \sum_i \frac{|C_i|}{N} \max_j Precision(C_i, L_j)$ where $Precision(C_i, L_j) = \frac{|C_i \cap L_j|}{|C_i|}$ and $N = \sum_i |C_i|$. Inverse purity is defined as$IP = \sum_i \frac{|L_i|}{N} \max_j Precision(L_i, C_j)$ where $N = \sum_i |L_i|$.

Quality of the output lists is evaluated by the F-measure, which is the harmonic mean of purity and inverse purity: $F = \frac{1}{1/P + 1/IP}$.

We did not used B-cubed evaluation measures[16] because B-cubed is an element-wise definition, which calculates correctness of all rows in the corpus, indicating that we would have to consider rows that have no headings, for which no clusters are generated. B-cubed measures are developed as a metric that works for soft-clustering, whereas our task can be regarded as hard clustering, in which P-IP measures work well.

We used *micro-averaged* and *macro-averaged f-measures* for cluster evaluation. Macro-averaged f-measure compute f-value for each document that has any repeated patterns (i.e., 26 documents in the test set) and average all the f-values. However, micro-averaged f-measures regard all 40 documents in the test set as one document, and calculate P, IP, and F on this one large document. Thus, we can evaluate how each method *does not* extract unnecessary lists from documents with no repeated lists by using a micro-averaged f-measure.

6.2 Baseline

We use the baseline algorithm that uses some heuristic rules to extract subdocuments. We test several configurations (e.g., what header tags are used for extraction, whether rows with $|r| = 1$ are extracted as headings, etc.) and select the one that performed the best on the test set. This baseline algorithm selects heading rows among all rows (except the ones discarded by the sentence row finder) using following heuristic rules.

First, it uses "header tag heuristics". For example, if the row is in an `<h2>` tag, we assume that the row is a heading that modifies the following blocks until the next `<h2>` or larger headers (`<h1>` in this case) appear. Header tags `<h1>`, `<h2>`, `<h3>`, and `<h4>` are used in this heuristics.[3]

Second, it uses the *block number heuristics* which showed good performance in our preliminary experiments. Assume that $|r|$ is the number of blocks in the row $|r|$. If $|r| \geq 2$, the algorithms regard r as heading row (we assume that this row is bracketed by `<h8>`, which is smaller than all other `h` tags.) If $|r| = 1$ and r is not a sentence row, we assume r is bracketed by `<h7>`, which indicates that it will be the heading of the next rows (if the next row has more than one blocks.)

[2] It is set to the end position of the document for the last subdocument in the list.

[3] We also used `<h7>` and `<h8>` generated by the block number heuristics described below.

Table 1. Averaged F-measure (%) for Each Method

Method	w/o no-repeat (26 docs.)	w/ no-repeat (40 docs.)
micro-averaged		
Proposed	50.23	47.68
Baseline	46.93	42.42
macro-averaged		
Proposed	49.63	—
Baseline	42.05	—

Note that this simple heuristics can extract many sub-documents in Figure 2 including "Age:25" and "TEL:+01-234-5678".

6.3 Results

We run our Gibbs sampling with 1000 initial iterations and 500 final iterations. Values of parameters $(\alpha_B, \alpha_E, \alpha_L, \alpha_N)$ were set to $(10, 100, 100, 1000)$ heuristically. We use the uniform distribution for each base distribution. Results were obtained by running Gibbs sampling 5 times and averaging all the averaged f-measure values.

Table 1 shows the results. Our algorithm outperformed the baseline algorithm by about 3.3 – 7.6 points. The performance gain of our algorithm in micro-averaged f-measure increased from 3.3 to 5.3 by using 14 "no-repeat" documents. This result suggests that our method works well in detecting "no-repeat" documents to avoid incorrect repeated lists.

Performance gain was mainly obtained by detection of heading blocks that could not be found by the baseline algorithm and detection of content blocks that could not be found by sentence row finder heuristics. However, the performance of our algorithm for documents with heading blocks that were easily detected by the baseline algorithm tended to be lower. We need an algorithm that takes the strength of both our method and the baseline method for better performance.

7 Conclusion

In this study, we proposed a probabilistic model for document structures for HTML documents that uses Bayesian hierarchical modeling. Our model can simultaneously manage both local coherence and global tendencies of layout usage, thanks to hierarchical modeling and cache effects obtained by integrating out of distributions. Experimental results showed that document structures obtained by our model were better than those obtained by the heuristic baseline method. For future study, we are keen to improve the performance of our method by, for example, using larger data sets to obtain more reliable knowledge about layout usage, or using more sophisticated methods to obtain maximum-likelihood states for our model.

References

1. Miao, G., Tatemura, J., Hsiung, W.P., Sawires, A., Moser, L.E.: Extracting data records from the web using tag path clustering. In: Proceedings of WWW 2009, pp. 981–990 (2009)
2. Liu, B., Grossman, R.L., Zhai, Y.: Mining data records in web pages. In: Proceedings of KDD 2003, pp. 601–606 (2003)
3. Chung, C.Y., Gertz, M., Sundaresan, N.: Reverse engineering for web data: From visual to semantic structures. In: ICDE (2002)
4. Yang, Y., Zhang, H.: HTML page analysis based on visual cues. In: Proceedings of the Sixth International Conference on Document Analysis and Recognition, ICDAR 2001 (2001)
5. Nanno, T., Saito, S., Okumura, M.: Structuring web pages based on repetition of elements. In: Proceedings of the Second International Workshop on Web Document Analysis, WDA 2003 (2003)
6. Mukherjee, S., Yang, G., Tan, W., Ramakrishnan, I.: Automatic discovery of semantic structures in HTML documents. In: Proceedings of the Seventh International Conference on Document Analysis and Recognition, ICDAR 2003 (2003)
7. Crescenzi, V., Mecca, G., Merialdo, P.: ROADRUNNER: Towards automatic data extraction from large web sites. In: Proceedings of the 27th International Conference on Very Large Data Bases (VLDB 2001), pp. 109–118 (2001)
8. Chang, C.H., Lui, S.C.: IEPAD: Information extraction based on pattern discovery. In: Proceedings of the 10th International WWW Conference (WWW 2001), pp. 681–688 (2001)
9. Nguyen, C.K., Likforman-Sulem, L., Moissinac, J.C., Faure, C., Lardon, J.: Web document analysis based on visual segmentation and page rendering. In: Proceedings of International Workshop on Document Analysis Systems (DAS 2012), pp. 354–358. IEEE Computer Society (2012)
10. Hu, Y., Xin, G., Song, R., Hu, G., Shi, S., Cao, Y., Li, H.: Title extraction from bodies of HTML documents and its application to web page retrieval. In: Proceedings of the 28th Annual International ACM SIGIR Conference (SIGIR 2005), pp. 250–257 (2005)
11. Tatsumi, Y., Asahi, T.: Analyzing web page headings considering various presentation. In: Proceedings of the 14th International Conference on World Wide Web Special Interest Tracks and Posters, pp. 956–957 (2005)
12. Cai, D., Yu, S., Wen, J.R., Ma, W.Y.: Extracting content structure for web pages based on visual representation. In: Zhou, X., Zhang, Y., Orlowska, M.E. (eds.) APWeb 2003. LNCS, vol. 2642, pp. 406–417. Springer, Heidelberg (2003)
13. Weninger, T., Fumarola, F., Barber, R., Han, J., Malerba, D.: Unexpected results in automatic list extraction on the web. ACM SIGKDD Explorations Newsletter 12(2), 26–30 (2010)
14. Teh, Y.W., Jordan, M.I., Beal, M.J., Blei, D.M.: Hierarchical dirichlet processes. Journal of the American Statistical Association 101(476), 1566–1581 (2006)
15. Artiles, J., Gonzalo, J., Sekine, S.: The semeval-2007 weps evaluation: Establishing a benchmark for the web people search task. In: Proceedings of the Workshop on Semantic Evaluation (SemEval 2007) at ACL 2007, pp. 64–69 (2007)
16. Artiles, J., Gonzalo, J., Sekine, S.: Weps 2 evaluation campaign: overview of the web people search clustering task. In: Proceedinsg of the 2nd Web People Search Evaluation Workshop (WePS 2009), 18th WWW Conference (2009)