Data Science and Engineering (2018) 3:101–114
https://doi.org/10.1007/s41019-018-0067-3

Exploiting Multi-Category Characteristics and Unified Framework to Extract Web Content

Jingwei Zhang[1,2] · Qian Wang[1,2] · Qing Yang[1,2] · Rui Zhou[3] · Yanchun Zhang[4]

Received: 4 April 2018 / Revised: 30 May 2018 / Accepted: 30 May 2018 / Published online: 7 June 2018

Abstract
Extracting web content is to obtain the required data embedded in web pages, usually including structured records, such as product information, and text content, such as news. Web pages use a large number of HTML tags to organize and to present various information. Both knowing little about the structures of web pages and mixing kinds of information in web pages are making the extraction process very challenging to guarantee extraction performance and extraction adaptability. This study proposes a unified web content extraction framework that can be applied in various web environments to extract both structured records and text content. First, we construct a characteristic container to hold kinds of characteristics related with extraction objectives, including visual text information, content semantics(instead of HTML tag semantics), web page structures, etc. Second, the above characteristics are integrated into an extraction framework for extraction decisions on different web sites. Especially, we put forward different strategies, path aggregation for extracting text content and HMM model for structured records, to locate the extraction area by exploiting both those extraction characteristics. Comparative experiments on multiple web sites with popular extraction methods, including CETR, CETD and CNBE, show that our proposed extraction method can provide better extraction precision and extraction adaptability.

Keywords Web extraction · Visual characteristics · Content semantics · Unified framework

1 Introduction

A vast number of websites and web pages produce large-scale and popular web content, which are making great contributions for data-driven applications and novel business modes. Before further exploitation of these web content presented in web pages, an extraction process should be started to conduct data preprocessing for constructing analysis applications on massive web content. The extraction process aims at discovering the concerned visual content in web pages, removing the surrounding noise and then separating them from web pages. However, websites use various HTML tags to organize and to present their content in a variety of formats, which constitutes the major obstacle for web content extraction. In addition, some extra information along with free web services, such as advertisements, recommended links, etc., increases the difficulty of identifying relevant content accurately and automatically. It is very valuable to establish effective methods and to provide clean web content for both existing and future big data applications.

Most of the existing extraction methods adopt an extraction mode of learning rules first and then extracting, a popular branch is to use supervised or semi-supervised methods to learn the extraction rules based on web page structures, such as DOM (Document Object Model) tree [1, 2] or XPath [3], and then to use the output rules to find the extracted objectives. However, the myriad web page templates adopted by different web sites, especially rich formats contributed by flexible and creative users, may

✉ Qing Yang
gtyqing@hotmail.com

1 Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China

2 Guangxi Cooperative Innovation Center of Cloud Computing and Big Data, Guilin University of Electronic Technology, Guilin 541004, China

3 Faculty of Science, Engineering and Technology, Swinburne University of Technology, Melbourne, Australia

4 Centre for Applied Informatics, Victoria University, Melbourne, Australia

cause the learned extraction rules failed. The adaptability is necessary for the contemporary extraction methods except for extraction precision. Both the diversity and the variability of web page structures brought by presentation requirements make a bigger extraction challenge. Compared with extraction mode of rule learning first, the new extraction mode should make an instant extraction decision automatically on the characteristics of the current web pages to address the above challenges.

In this work, our goal for web content extraction is to provide an effective and adaptive extraction decision without explicit extraction rules, which can work well on a wide variety of web sites covering plentiful web page templates and noise. First, we construct a characteristic container to hold different extraction characteristics, which is initialized by a novel visual content characteristic that can help to find some well-marked text nodes as starting points to enlighten the following extraction. And then we design different methods for extraction objectives to locate the extraction area, namely a path aggregation computation for extracting text content and a HMM model for extracting structured records. At last, we will refine the above processes since their results maybe include some noise for web content extraction. We exploit content semantics to remove those noise and to improve extraction performance, which is very different from the semantics of HTML tags. All the above components, including enriching extraction characteristics, locating extraction area, and refining intermediate results, are then integrated into an extraction framework for instant extraction decisions.

In summary, this study makes the following contributions,

- Designed a unified framework based on multi-characteristics of web pages to extract web content, including both structured records and text content, which allows to make instant extraction decisions on web pages' own characteristics without any explicit extraction rules and provides extraction adaptability.
- Defined two novel characteristics for extraction, namely visual text information and content semantics, especially, content semantics is very different from HTML tag semantics. The two kinds of characteristics are integrated with the DOM model of web pages to support accurate extraction decisions in different web sites. We also permit to extend different characteristics for extraction.
- Demonstrated the effectiveness of our method by conducting comprehensive and comparative experiments on multiple web sites, which cover different existential situations of web content.

The rest of this paper is organized as follows. In Sect. 2, we review the related work on web extraction. We present the detailed problem definition for web content extraction in Sect. 3. Section 4 describes the extraction framework and elaborates the extraction characteristics and the extracting process. In Sect. 5, we investigate the effectiveness of our method. Finally, we conclude our work in Sect. 6.

2 Related Work

Web content usually includes two different forms, the visual formal text(such as news, blogs, etc.) and structured records (such as product information, etc.), which do not include advertisements, links and other noise. Web content extraction was firstly put forwards by Rahman et al. [4], which has been playing an important role on data collecting and cleaning for big data applications. From the viewpoints of extraction technologies, there are three primary kinds of extraction strategies, rule-based extraction, vision-based extraction and semantics-based extraction.

Rule-based extraction, also named wrapper-induction extraction, often depends on the web page structure contributed by HTML tags, which considers a web pages as a DOM tree or a character stream. Valter et al. [5] proposed Roadrunner, which can establish wrappers described by regular expressions automatically on both the similarities and the differences in a group of web pages. Elemeleegy et al. [6] designed supervised method to construct extraction templates for product information. Furche et al. [7] studied the problems of both extraction robustness and noise resistant, but their method needs to annotate web pages manually and is not a completely automatic method. Reis et al. [8] put forwards a domain-oriented approach to extract news, whose core is to compute tree edit distance on the DOM model of web pages. Wu et al. [9] defined tag path edit distance and tag path ratios to extract news from web pages. Furthermore, they developed the relevancy between web content layouts and tag paths and exploited tag path features to extract web content [10]. Qiu et al. [11] proposed DEXTER, which can extract product specifications from web pages and discover some hidden attributes. Wu et al. [12] integrated a joint-training method with naive Bayesian classification to identify attribute candidates and then exploited the features of those attribute candidates to locate the areas holding goods information. Rule-based methods need to create different wrappers for different sites, once the web page templates changed, the learned rules will fail. In addition, the pure structure characteristics are not enough to deal with today's complex web environments well.

Vision-based extraction makes full use of web page layouts and other visual cues to improve extraction performance, which also often collaborates with web page structures, such as DOM tree. Cai et al. [13] proposed an

extraction algorithm named VIsion-based Page Segmentation(VIPS), which utilizes both the DOM tree and visual layout cues to segment documents and then to find those segmentations containing the required content. Based on the VIPS, Song et al. [14] constructed feature vector for block importance, including spatial features(such as position, size) and content features(such as the number of pictures and links), and considered the blocks with high rankings as extraction objectives. Fernandes et al. [15] also adopted the strategy of ranking the blocks of web pages by computing their weights to extract content. Sun et al. [16] demonstrated the efficiency of text density for content extraction, which introduces text density both under a single node and in a whole web page to work with DOM tree for more accurate extraction. Qureshi et al. [17] developed more statistical and formatting characteristics of web pages on DOM trees, such as text information and links density, font size, style and location, to establish a hybrid extraction model. The vision-based extraction technologies need to make careful observations of visual cues, especially, different formatting preferences often generate different visual features, whose applications still have some constraints.

Semantics-based extraction utilizes the known functions of HTML tags, namely the semantics of tags, to improve extraction performance. Peters et al. [18] integrated the semantics of several specific HTML tags for content extraction, such as ⟨class⟩, ⟨div⟩, ⟨$h1$⟩, etc. Both the flexible nested usage of HTML tags and the growing number of HTML tags, such as HTML5 tags, only permit us to exploit the semantics of a part of HTML tags.

Some further characteristics are also exploited for extracting web content. Ortona et al. [19] proposed WADaR, which collaborates joint wrappers with data repair to provide better extraction performance. Weninger et al. [20] proposed Content Extraction via Tag Ratios(CETR), which considers HTML tag ratios as an important feature and designs a clustering technique based on 2-dimension tag ratio to distinguish content and non-content areas in web pages. Uzun et al. [21] introduced a feedback mechanism between the decision tree learning for obtaining the extraction rules and the extraction precision on the specific rules, which aims at getting a tradeoff between extraction efficiency and effectiveness. [22] introduced user reviews to extract product attributes. Wong et al. [23] modeled the dependency between web page layout and structured data presentation into a probability graph to extract structured data. Zheng et al. [24–26] and Zhang et al. [27, 28] studied both keyword search and similarity computing for analyzing trajectory data, those query-based strategies can be also valuable for extracting web content.

Web environments have been in evolution; it is still a challenging work to develop an effective and strongly adaptable content extraction framework. We should exploit further characteristics to respond to the extraction challenges caused by diversified presentation requirements and flexible user formatting and to design novel methods for more accurate and adaptive extraction. Weninger - et al. [29] also conducted in-depth discussions on the current web content extraction technologies in 2015, who believe that the evolution of web site formats and practices are bringing big challenges for content extraction, both rule-based extraction methods and extractors based on heuristic features should be re-examined, new extraction methods should be designed to address future extraction problems.

3 Problem Statement

For collecting web content, we focus on two kinds of web pages in this work, the first kind are those pages holding news, blogs, etc. whose core contents are a few paragraphs of text describing a complete story, and the other kind are those pages holding structured records, such as a e-commence web page presenting product information. Usually, a web page can be considered as a character stream or a DOM tree for further processing. To make full use of the structural characteristic of web pages implied by HTML tags, we define a web page on the viewpoint of DOM tree. The DOM tree of a web page is composed of two types of nodes, branch nodes showing web page organization and leaf nodes holding web page information. The values of branch nodes are just HTML tags, the values of leaf nodes usually text or some other visual information, such as links, advertisements, etc.

In a DOM tree, a tag path is a series of ordered HTML tags, which starts from the root node to one leaf node in DOM tree. An effective text node means that the text node located by a tag path holds the required content that should be extracted. A web page can be defined by a set of pairs, $\langle \text{path}, \text{content} \rangle$, where path corresponds to a tag path, and content is just the information held by the leaf node located by path. Figure 1a and b illustrate an excerpt of DOM tree and the corresponding definition for the web page.

An extraction operation is to make an extraction decision for each pair, a boolean value will be attached for each pair. For example, if the content located by path is a required content, the pair will be transformed into $\langle \text{path}, \text{content}, \text{TRUE} \rangle$, otherwise $\langle \text{path}, \text{content}, \text{FALSE} \rangle$. Given a group of web pages, which can be transformed into a series of pairs $\{p_1, p_2, \ldots, p_n\}$, the content extraction process aims at expanding each pair $p_i = \langle \text{path}_i, \text{content}_i \rangle$ into a triple $\text{ep}_i = \langle \text{path}_i, \text{content}_i, \text{decision}_i \rangle$ and then

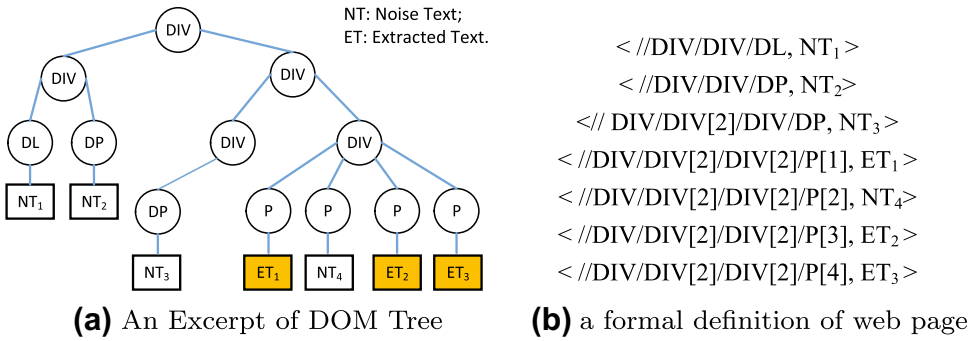

Fig. 1 An excerpt of DOM tree and the definition of web page. **a** An excerpt of DOM tree, **b** a formal definition of web page

outputs those triples with decision = 1, namely $\{ep_i | ep_i = \langle path_i, content_i, decision_i \rangle \wedge decision_i = 1\}$. We will construct a unified framework to extract both text content and structured records embedded in various web pages.

4 Unified Extraction Framework for Web Content

In this section, we will elaborate a three-phase extraction framework, which is comprised of modeling and preprocessing web pages to discover extraction characteristics held in an extensible characteristics container, identifying the extraction area based on the above characteristics, and refining the extraction decisions by semantics similarity. Before extraction, since HTML tags contribute greatly for web page structure except for making information visual, all web pages will be modeled into DOM trees for easy utilization of structural characteristics, and some redundant parts are also pre-pruned for simplifying later computation. Three kinds of characteristic, including visual punctuation mark characteristics, Web page structure characteristic and content semantics similarity, are defined and refined, which are then applied for the following extraction decisions.

Figure 2 presents the proposed content extraction framework and the input/output in each phase, in which the green dashed arrows represent input or output, the red dashed lines represent that the paths do not locate the required content and should be removed, all leaf nodes with orange filling in DOM trees represent the required content. Figure 2c presents the three phases of the extraction process. When a web page shown in Fig. 2a is input into the extraction framework, the required content will be output in the third phase, Fig. 2e shows the set of required triples and illustrates the final DOM tree, in which all paths marked with red dashed lines are removed, those remaining paths can be used to extract the required content. Figure 2b and d demonstrate the internal input/output and the corresponding changes of web page.

4.1 Constructing DOM Tree and Web Page Preprocessing

Except for presentation functions, HTML tags also provide valuable structural information for web pages, which are very helpful to improve extraction performance. The nested structure of HTML tags in web pages can be interpreted into a tree structure, which is named DOM model. We use Jsoup parser [30] to interpret web pages into DOM trees. DOM tree is the logical model of web pages, which allows web pages to be processed easily in memory. DOM model transforms each pair of HTML tags into a subtree, such as ⟨DIV⟩ and ⟨/DIV⟩. A traversal operation can be exerted on DOM tree to access all branch and leaf nodes, and then output ⟨path, content⟩ pairs. In addition, we also use regular expressions to remove those unrelated tags and parts from web pages, such as subtrees covered by ⟨script⟩ and ⟨/script⟩, which are not relevant to the required content. The pre-pruning will simplify the web pages, which is demonstrated in Fig. 2b, c and d.

4.2 Computing Visual Characteristics on Punctuation Marks Appearance

Text density, layouts of web pages and some other visual characteristics have been developed to improve extraction performance, but they have their own constraints. For example, text density can only work well for long text, in fact, from the structural viewpoint of web pages, a paragraph of text is often organized by multiple HTML tag pairs for the purpose of presentation, which separates a paragraph of text into several short text block. Here, we will develop the intrinsic characteristic of text content to improve extraction performance.

Punctuation marks are necessary for web content to describe some specific things. Both a topic-related web page holding several paragraphs of text to tell a complete story and a web page describing product information use punctuation marks for the readability of web content. According to the large number of observations and statistics on web pages, punctuation marks present apparent

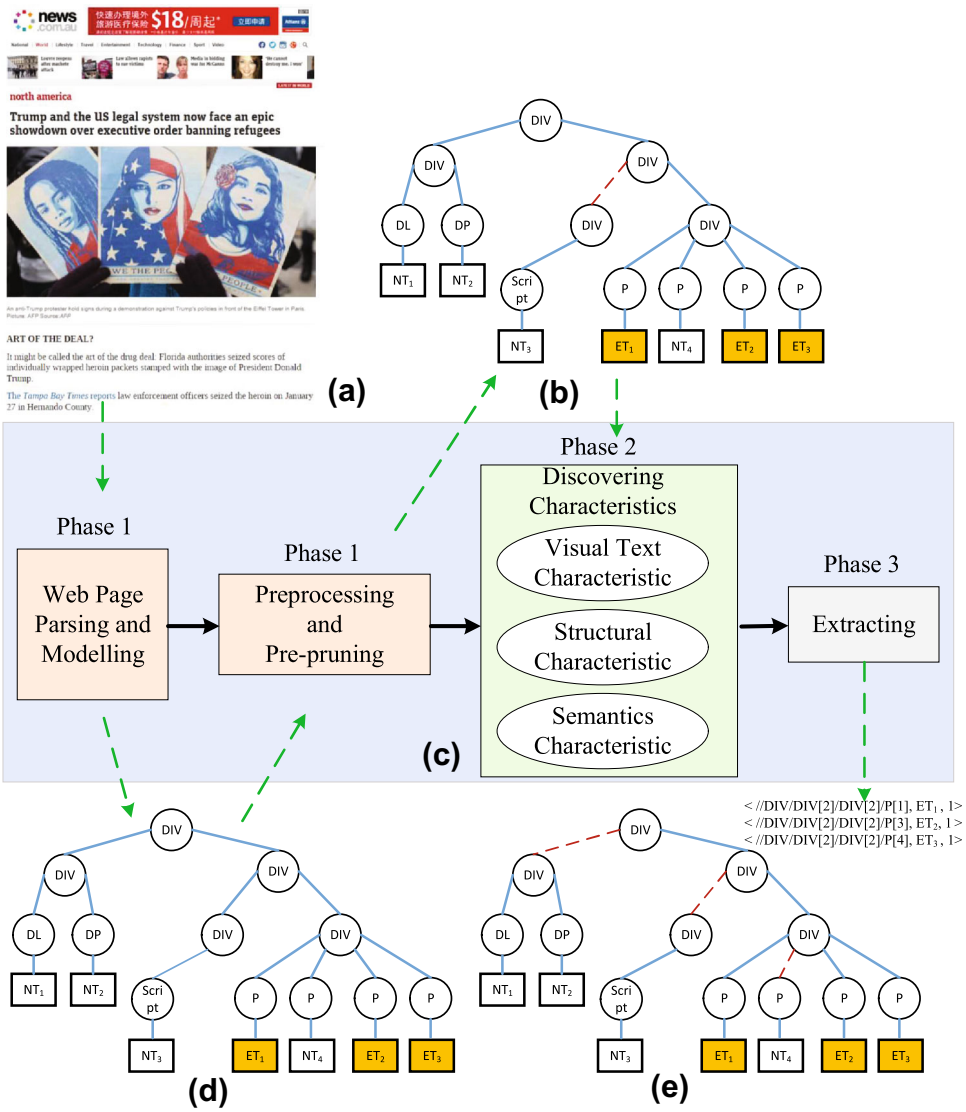

Fig. 2 Content extraction framework and its input/output

usage difference between the required content and those areas holding noise, such as navigation, recommendation links, etc. The required content in web pages are usually accompanied with a large number of punctuation marks since they are presented for easy reading. Though noise, such as navigation panels, advertisements, etc, may be presented in text, they contain few punctuation marks. Figure 3 makes an illustration about the punctuation mark characteristics in different areas of a web page. Figure 3a–c shows parts of the required content, navigation and recommendation links, respectively, and their corresponding DOM trees, punctuation marks are more used in the area holding the required content, but rarely appear in those noise areas. Web pages holding structured records also present the similar characteristic, for example, ':' is used to separate an attribute and its value. The above observation is an important cue for extraction; we will use the punctuation mark characteristics accompanying with the required content to identify those effective text nodes with distinctive punctuation marks.

In order to identify effective text nodes with the help of the visual punctuation mark characteristics, we define the Visual Value of Text Characteristics (vvtc) in Formula 1 to weight the text characteristics of each node in DOM tree, which combines the text length and the punctuation mark weight for each node.

$$\text{vvtc}_a = \frac{\text{text\_length}_a}{\text{nodes\_num}} * \frac{\text{punc\_num}_a}{\text{nodes\_num}} \quad (1)$$

Here, a is a leaf node in the DOM tree, text_length_a is the length of the text held by the node a, punc_num_a is the total number of various punctuation marks in the above text, nodes_num is the number of all leaf nodes in the DOM tree. A node with high vvtc value indicates its text should be extracted. All nodes can be sorted in descend order on their vvtc value, the top-k nodes can be kept. For

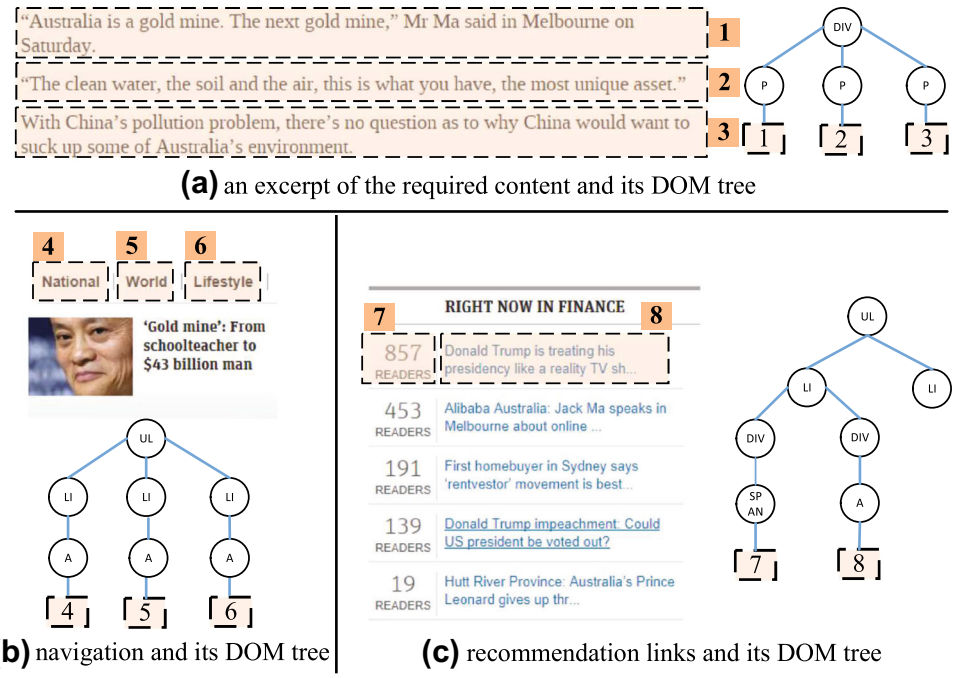

Fig. 3 Punctuation marks in different areas of web page. **a** An excerpt of the required content and its DOM tree, **b** navigation and its DOM tree, **c** recommendation links and its DOM tree

example, the vvtc values of node (1)–(8) in Fig. 3 are 1.25, 1.59, 1.56, 0, 0, 0, 0, 0.17, respectively. Node (1)–(3) will be extracted if top-3 nodes are reserved.

The vvtc value cannot be considered as a direct proof for effective text nodes. It can only tell that a node is an effective text node with a high possibility, especially when some nodes with high vvtc value are organized together in a DOM tree, just like those nodes numbered 1–3 in Fig. 3a. In fact, the visual characteristic exerts a strict operation to discover text content, which will cause it cannot cover the complete text content. We use a container to hold all characteristics, which ensure the extensibility of extraction characteristics.

4.3 Identifying Extraction Areas

This section is to identify the accurate areas holding the required content by the above induced characteristics, we will design different mechanisms to extract text content and to extract structured records since they have often different features when presented in web pages.

4.3.1 Aggregating Paths to Discover Text Content Areas

For extracting text content, the visual text characteristic only outputs those tag paths locating text content with apparent punctuation marks, etc. Considering the DOM model, a paragraph of text content in web page is often organized in several different text nodes according to the formatting and presentation requirements, which prevents the visual characteristic from discovering all content that should be extracted. For example, the punctuation mark characteristic does not apply for text that is given as special presentations since it is very short and does not have any punctuation marks, such as using $\langle h \rangle$ for highlights. Especially, the required text nodes are usually neighbors or siblings in DOM tree though it is possible for them to be separated by some other nodes, such as nodes holding image titles.

Since those text nodes holding a paragraph of text are neighbors or nearby, which implies that tag paths locating a paragraph of text have a common prefix, those text nodes nearby the discovered tag paths may also be the expected ones. We can extend the discovered results on the visual text characteristic to cover the whole content area. Our strategy is to aggregate tag paths to discover the maximum common path prefix and then to locate broad content areas. Here, a block of content area is just a subtree constructed by some tag paths with a common prefix.

Those tag paths discovered in setction 4.2 need to be aggregated for locating the whole content areas. Supposing X and Y are two tag paths, represented as $X = \langle x_1, x_2, \ldots, x_i \rangle$, and $Y = \langle y_1, y_2, \ldots, y_j \rangle$, x_i and y_j are tags, the aggregation of X and Y can be defined in Formula 2. $\text{pcs}(X, i, Y, j)$ holds the length of common substring between X and Y. The area covered by the resulted tag path, which is constructed by the first $\text{pcs}(X, i, Y, j)$ tags of X, will be considered as content area.

$$\mathrm{pcs}(X,i,Y,j) = \begin{cases} 0 & \text{if } (i=0) \text{ or } (j=0), \\ \mathrm{pcs}(X,i-1,Y,j-1)+1 & \text{if } (i,j>0) \text{ and } (x_i=y_j), \\ \max(\mathrm{pcs}(X,i,Y,j-1),\mathrm{pcs}(X,i-1,Y,j)) & \text{if } (i,j>0) \text{ and } (x_i\neq y_j). \end{cases} \quad (2)$$

This above extraction operation aims at covering those effective text content with no punctuation marks. Compared with extraction process in Sect. 4.2, tag path aggregation is a generalized process to locate the content areas and in fact loosens the results, which causes it possible to include some noise in the content areas. For example, image title presented in the middle of a paragraph of text will be extracted in this phase.

4.3.2 Constructing HMM Model to Identify Areas Holding Structured Records

In this section, we will focus on constructing a HMM model by the above extraction characteristics and the DOM model, which will be applied to find the extraction area holding structured records. In addition to some delimiters are considered in the characteristics container, such as ':', an attribute dictionary are also introduced to improve extraction performance, which holds those popular attribute names, such as 'goods name', 'price', etc. All words in this dictionary will also be a kind of characteristics for extraction.

A. Assigning Weight Based on the Extraction Characteristics

First, we introduce the weight definition for non-leaf nodes in the DOM tree, Formula 3 presents the characteristic weight of the ith non-leaf node, $\mathrm{WVC}(i)$. All characteristics in the container are used to compute the weights of non-leaf nodes, a bottom-up strategy and a $tf*idf$-similar method are applied for the weight computation.

$$\mathrm{WVC}(i) = \sum_{k=1}^{m} n_k * \log\frac{N}{n_k} \quad (3)$$

In Formula 3, n_k denotes the number of those leaf nodes that are the descendants of the ith node and satisfy the kth characteristic, m is the number of the characteristics held in the characteristic container. N denotes the number of all leaf nodes in the whole DOM tree. A high value of $\mathrm{WVC}(i)$ means that the ith node is important for extraction. A bottom-up strategy is efficient to compute the weights of all non-leaf nodes since the n_k of each parent node can be obtained by summing the corresponding values of all its child nodes.

B. Constructing HMM Model on DOM Tree

Since the areas holding the required web content are located by a path composed of a sequence of HTML taggs, we assume that the status of the current node in a DOM tree is only decided by its parent node and then can construct a HMM model based on the DOM tree. We use a triple, $\lambda = (\pi, A, B)$, to represent the HMM model, in which π is the initial distribution of node statuses, A corresponds to the transition probability distribution of node statuses, and B shows the weight probability distribution between the observation statuses and the hidden statuses. The hidden statuses are decided by the node number when traversing the DOM tree top-down. The transition probability decides how to allocate the characteristic weight from a parent node to a child node, which is initialized in a equi-probability mode. An observed sequence is a series of HTML taggs from the root node to a leaf node.

We introduce the Viterbi algorithm λ to compute the extraction area, which applies dynamic programming to find the most probable hidden state sequence. λ and an observed state sequence O are its input and a hidden state sequence is its output. The output can locate the areas holding the required records, and the detailed process is presented as following.

Input: $\lambda = (\pi, A, B)$, $O = (O_0, O_1, \ldots, O_t)$, O_i is HTML tag of the ith node;

Ouput: the most probable path $I = \langle i_0^*, i_1^*, \ldots, i_m^* \rangle$, i_j^* is the number of the jth node;

- *Phase 1* Initialization.
  At time t_0 (just the t_0 level of a DOM tree), we have $\delta_0(0) = \pi_0$ and $\psi(0) = 0$. Here, $\delta_j(i)$ denotes the weight of the node with HTML tag O_i at time t_j, $\psi(j)$ denotes the number of the node with the highest weight at time t_j. In our case, the node with the tag **body** is the initial node, denoted by O_o, whose state is denoted as $\delta_0(0)$ at time t_0. At the same time, the node with the tag body is unique at time t_0, which is just the node with the highest probability, namely $\psi(0)$.
- *Phase 2* Iteration.
  At time t_1, $\delta_1(i) = \pi_0 a_{0i} * b(O_i)$, $\psi(1) = \mathrm{argmax}_i(\delta_1(i))$. Here, a_{0i} corresponds to the transition probability from O_0 to O_i, $b(O_i)$ is the weight of the node with HTML tag O_i. From time t_2 to t_x, we have $\delta_k(i) = \pi_j a_{ji} * b(O_i)$ and $\psi(k) = \mathrm{argmax}_i(\delta_k(i))$, $2 \le k \le x$. In which, a_{ji} denotes the transition probability (also weight allocation) from jth node to the ith node, $\psi(k)$ holds the number of the node with the maximum weight in current time.
- *Phase 3* Termination.
  At time t_m, if $\psi(m) \ge \psi(m+1)$, we will terminate the computation process since we have reached the boundary of the extraction area. Now, the hidden state sequence can be output as a maximum probability path, namely
  $I = \langle \psi(0), \psi(1), \ldots, \psi(m) \rangle = \langle i_0^*, i_1^*, \ldots, i_m^* \rangle, 0 < m \le x.$

4.4 Summarizing Content Semantics and Computing Similarity

In general, both the title of web page and the content covered by the tag ⟨description⟩ summarize the primary meaning of the whole text content, and both of them play an important role in revealing the content of web page. In order to refine extraction results, especially for extracting text content, we combine SimHash [31] and Hamming distance to compute the similarity between content area and web page title as well as the content covered by the ⟨description⟩ tag to determine whether the area covered by the output tag path in the second stage is the whole content area or should remove some noise.

Firstly, we use NLPIR word segmentation system [32] to get keywords of each text node in the content areas, and then the following process is exerted on those keywords to summarize text content, which is responsible for transforming a paragraph of text into a feature vector. Figure 4 illustrates the whole process.

(a) computing the weight of keywords. All keywords get their weight in the whole text area by TF–IDF presented in Formula 4, where k represents the k_{st} keyword, N is the number of text paragraphs in the text area, N_K is the number of text paragraphs containing k, L is an empirical constant. And then to construct $(\text{keyword}, \text{weight})$ pair set.

(b) computing hash value of keywords. SimHash is applied to get the hash value of each keyword, and then transform $(\text{keyword}, \text{weight})$ into $(\text{hash}, \text{weight})$.

(c) constructing feature vectors of keywords. For each pair $(\text{hash}, \text{weight})$, check each bit of hash, and use weight to substitute 1, and $-\text{weight}$ for 0.

(d) Outputting the feature vector of content area. Summing all vectors of keywords into a final vector. For each element of the final vector, if its value is greater than 0, then use 1 to replace it; otherwise, 0 is deployed.

$$\text{weight}(k) = \text{TF}_k * \text{IDF}_k = \text{TF}_k * \left[\log\left(\frac{N}{N_k}\right) + L\right] \tag{4}$$

Repeating the above process for the concatenation of the title of web page and the content covered by the ⟨description⟩ tag to get the final vector by SimHash. Then, Hamming distance is adopted to compute the similarity between two final vectors corresponding to web title as well as the content covered by the ⟨description⟩ tag and each paragraph of text belonging to a leaf node, respectively, which will tell how many bits are not consistent in the two final vectors. Apparently, a distance value in the limited range indicates that the paragraph of text has great relevance with web title as well as the content covered by the ⟨description⟩ tag, which should be extracted. A parameter threshold for Hamming distance will be designated for the extraction decisions, which will be discussed in Sect. 5.2.

5 Experiments

In this section, we will design and conduct experiments on our collected real data sets to investigate the effectiveness of the proposed extraction method. For extracting text content, we will firstly discuss the threshold parameters for the chosen characteristics, and provide performance comparison with the popular extraction methods, CETD [16] and CETR [20]. For extracting structured records, we will provide performance comparison with CNBE [12], a semi-supervised method applying naive Bayesian classification to extract pairs of attributes and values.

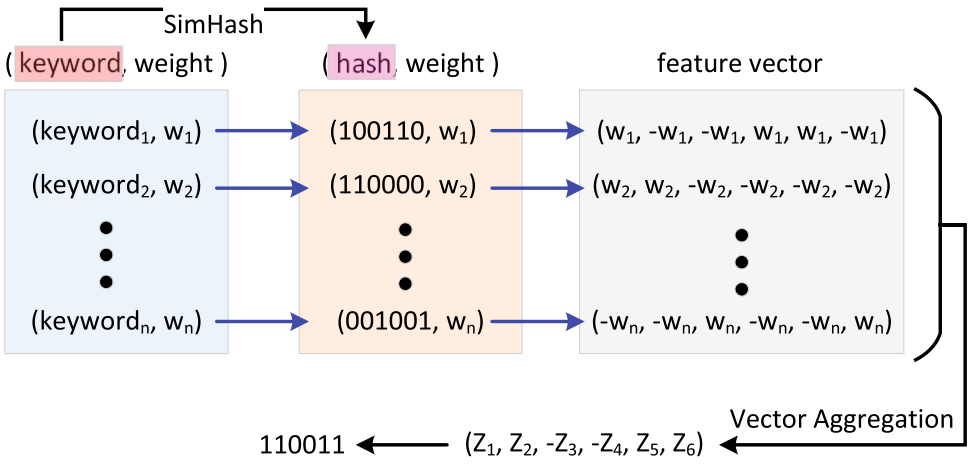

Fig. 4 Summarizing content semantics

5.1 Experiment Setting and Datasets

We use Java to implement the proposed extraction method, all operations on DOM trees are supported by DOM API in Java. Jsoup [30] Parser is adopted to patch the missed HTML tags and to interpret web pages into DOM models since it provides very convenient APIs and strong robustness for web pages with complex structures. All experiments run on the following experimental environments, Intel Xeon CPU E3@3.30 GHz and 8 GB RAM.

We established two real data sets by crawling a large number of web pages, which are used, respectively, to verify the extration performance of different kinds of web content, namely text content and structured records. Tables 1 and 2 list the URLs of those web sites and the corresponding abbreviations as their following symbols. For verifying the performance of text content extraction, we crawled web pages from 10 different web sites listed in Table 1, which includes 100 different web pages for each web site and covers different web environments. For verifying the performance of structured record extraction, we crawled web pages from three popular electronic commence sites listed in Table 2, the data set covered 8 categories, including computer, clothes, shoes, bags, jewel, furniture, book and food, and each category consists of 100 pages. All web pages in the data sets are firstly interpreted by Jsoup and then are used pre-defined regular expressions to remove irrelevant HTML tags, finally the preprocessed web pages are input into our extraction framework to extract the required content.

We use a trivial and boring process, manual XPath expressions combined with artificial confirmation, to extract all the required content accurately and to establish our ground truth for experimental evaluation. Precision($\frac{|ECS|}{|EAS|}$), Recall($\frac{|ECS|}{|GS|}$) and F1-score($\frac{2*|ECS|}{|EAS|+|GS|}$) are used as evaluation metrics. Here, EAS denotes the set of the extraction results from the data set by the proposed method, namely all extracted sentences for text content extraction or all records holding pairs of attributes and values for structured record extraction. ECS denotes the set of the required content or records in the extraction result. GS denotes the set of sentences or records existing in the ground truth.

5.2 Parameter Analysis

For better extraction performance, we introduce two threshold parameters, T_1 for the Visual Value of Text Characteristics and T_2 for Hamming distance. Especially, T_1 is defined as $\frac{\text{vvtc}_a}{\text{MaxVVTC}}$, where MaxVVTC is the maximum value of all vvtc values, vvtc_a denotes the vvtc value of the specific node a. A good threshold choice will enhance the extraction results, especially for text content extraction.

We design a group of experiments on the data set listed in Table 1 to reveal the relationships between the thresholds and the extraction performance, whose results are presented in Figs. 5 and 6. The experimental results in Fig. 5 show T_1 should be between 0.6 and 0.8 for a good extraction performance. Figure 6 tells that T_2 should be set to 3 for a tradeoff between precision and recall. In the following experiments, T_1 will be fixed at 0.8 and T_2 at 3.

5.3 Experimental Results and Analysis for Extracting Text Content

For verifying the effectiveness on text content extraction, we executed the proposed method (abbr. as VSSE) on the data set listed in Table 1 and compared its performance with the popular extraction methods, CETD [16] and CETR [20]. Our extraction method made a good extraction results, all performance indexes are above 90%, which proves its adaptability for different web environments. The overall extraction performance comparison on all web sites with CETD and CETR are presented in Figs. 7, 8, 9 and 10.

Figure 7 presents the precision comparison, the extraction precision contributed by our method has fully dominated CETR, which is increased by 20% for almost each web site, even up to 35% for some individual web site. For extraction recall, our method and CETR have similar performance, which are presented in Fig. 8. CETR computes the tag ratios per line in web pages, and then extracts content on the basis of clustering, which often brings a big coverage for the required content to cover some noise, this is a key reason for CETR to have a different performance between its precision and recall. The characteristics from visual punctuation marks and the web page structures

Table 1 Data set 1

No.	Website url	Pages	Abbr.	No.	Website url	Pages	Abbr.
1	http://news.ifeng.com	100	W_IF	6	http://news.cctv.com	100	W_CT
2	http://news.163.com	100	W_163	7	http://www.xinhuanet.com	100	W_XH
3	http://news.sina.com.cn	100	W_SN	8	http://www.chinanews.com	100	W_CN
4	http://news.qq.com	100	W_TC	9	http://news.sohu.com	100	W_SH
5	http://www.huanqiu.com	100	W_HQ	10	http://www.81.cn	100	W_81

Table 2 Data set 2

No.	Website url	Pages	Abbr.	Categories	Note
1	http://book.dangdang.com	800	DD	Computer, clothes, shoes, bag, jewel, furniture, book, food	100 pages each category
2	http://www.yhd.com	800	YHD		
3	http://www.jd.com	800	JD		

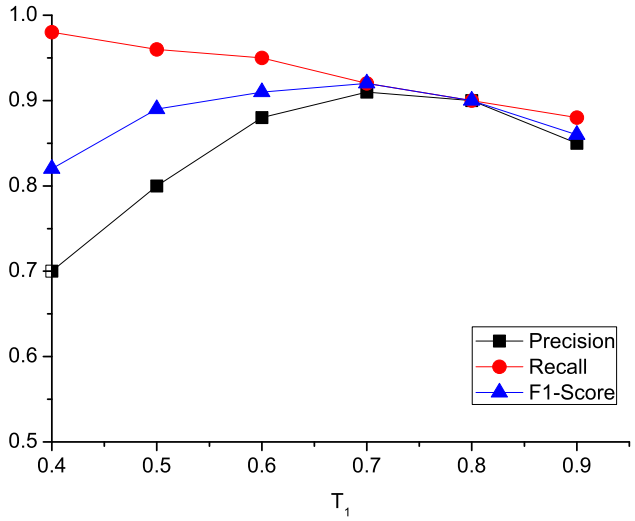

Fig. 5 Recall, precise and $F1$-score on different T_1

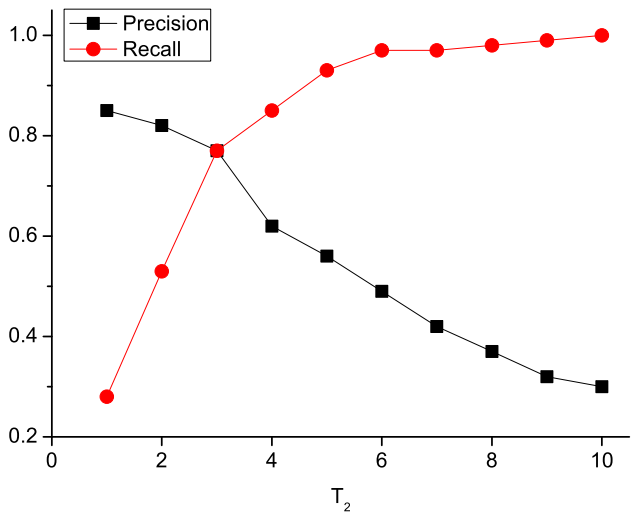

Fig. 6 Recall and precision on different T_2

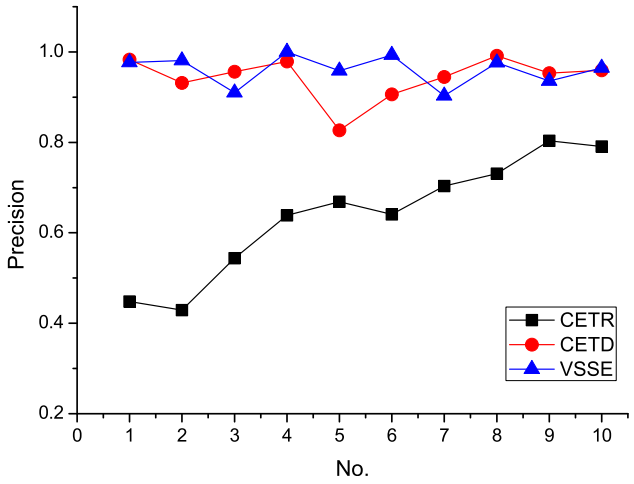

Fig. 7 Precision comparison

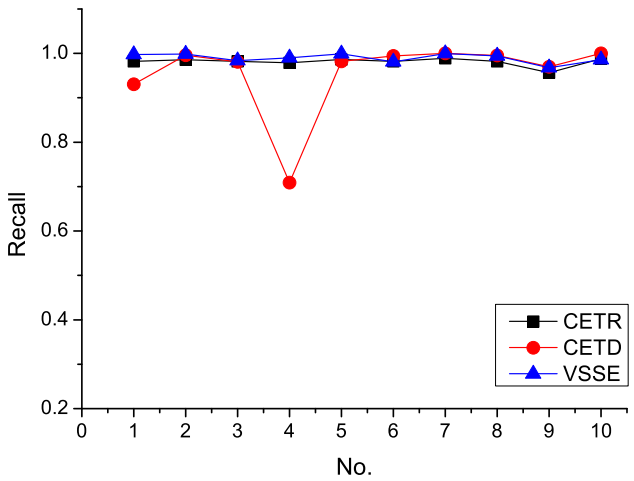

Fig. 8 Recall comparison

adopted by our method have the similar function with CETR, but a further extraction decision by content semantics similarity on the extracted items enhances the extraction precision, which also finally contributes to a steady performance on F1-score, the corresponding results are compared in Fig. 9. Compared with CETD, our method presents a better stability on recall though it has a slim advantage on precision, which only won by 2% on average precision. CETD considers the text density as a primary characteristic for extraction decision, the extraction results will present a big error when noise density is over text density in some nodes, such as the No.4 in Fig. 8.

From the experimental results presented in Figs. 7, 8 and 9, we can also see that the extraction performance of our method are very steady on different web sites, but the performance of both CETD and CETR are influenced by the different environments of web sites. Figure 10 makes a

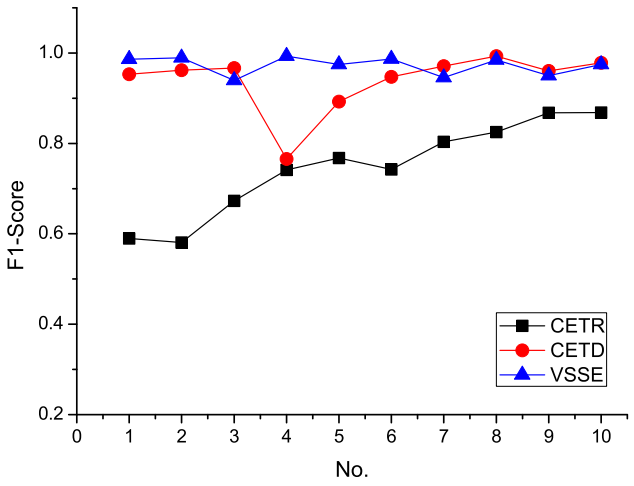

Fig. 9 *F*1-Score comparison

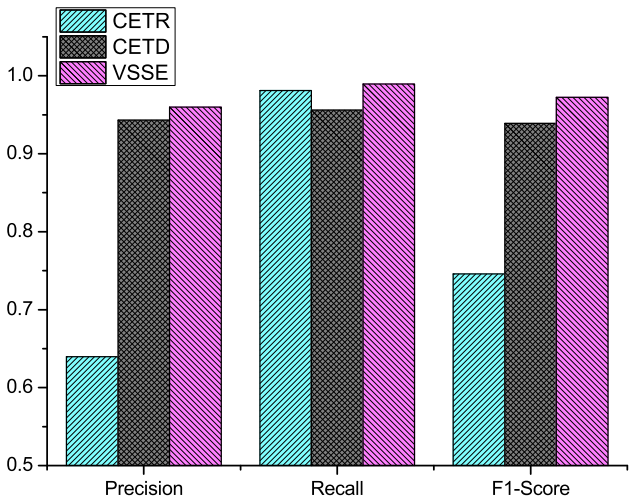

Fig. 10 Overall performance comparison

more intuitive presentation for their extraction performance by the overall performance on the whole data set. Our method works better with good adaptability on different web environments.

5.4 Experimental Results and Analysis for Extracting Structured Records

In this section, we designed experiments to verify the effectiveness of our proposed method on extracting structured records. A semi-supervised extraction method [12], abbr. as CNBE, is introduced for performance comparison. We crawled web pages holding different categories of goods to cover different web page templates and to verify the extraction adaptability of our proposed method.

Figures 11, 12 and 13 present the average extraction performance on the three data sets listed in Table 2. Our proposed method presents both high precision and good

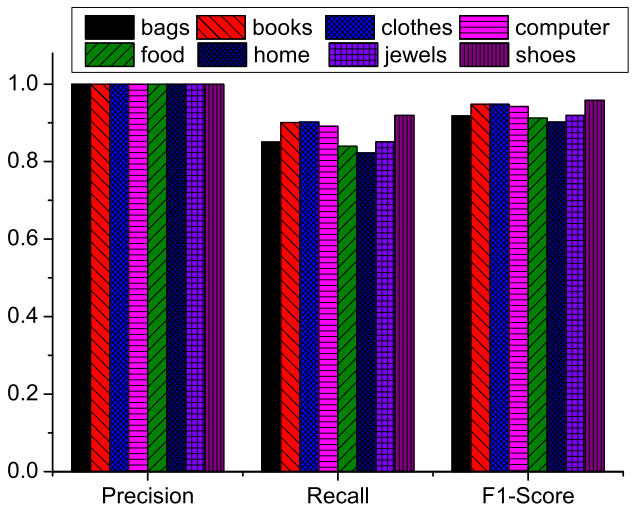

Fig. 11 Extraction performance on http://www.dangdang.com

adaptability on extraction performance though those web pages are from different web sites and holding various goods. Nearly, in all experiments, our method presents a perfect performance on extraction precision, which is contributed by both the HMM model and the reasonable characteristics. The experimental results present small difference on recall, but most of them are above 91%. A special case is the book category of http://www.yhd.com whose recall is only 76%, which is because the set of characteristics make strict constraints on their extraction so that some pairs of attribute-value are missed. The above experiments also show that the set of characteristics constraints have been an important factors to balance precision and recall.

Figures 14, 15 and 16 provides an extraction performance comparison with CNBE. Our method outperforms CNBE in all experiments on extraction precision and recall on the three data sets listed in Table 2. Figure 17 presents a comprehensive comparison view. Though the two methods contribute more than 98% precision, our method makes a better improvement on recall, from 85% of CNBE to 93% of our method, especially we only apply a single set of characteristics on all web pages. The integration between those effective extraction characteristics and the HMM model makes a great contribution to identifying the extraction area accurately, which brings a high recall when ensuring less unexpected extraction results.

6 Conclusions

This study contributes a novel content extraction method on various web sites, including extracting web text content and extracting structured records. A flexible characteristics container and a unified extraction framework are proposed

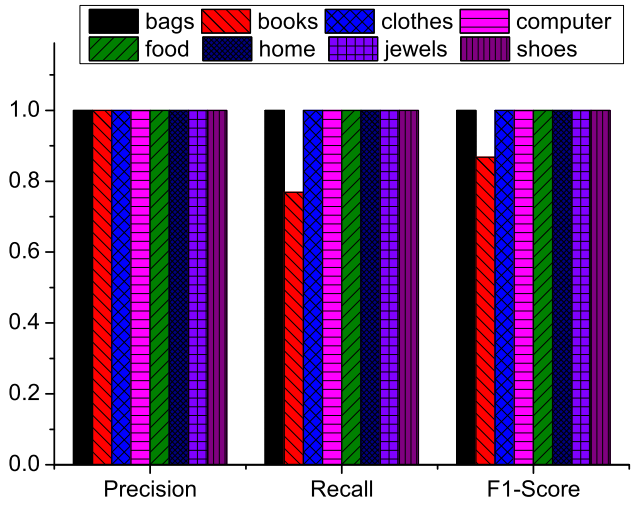

Fig. 12 Extraction performance on http://www.yhd.com

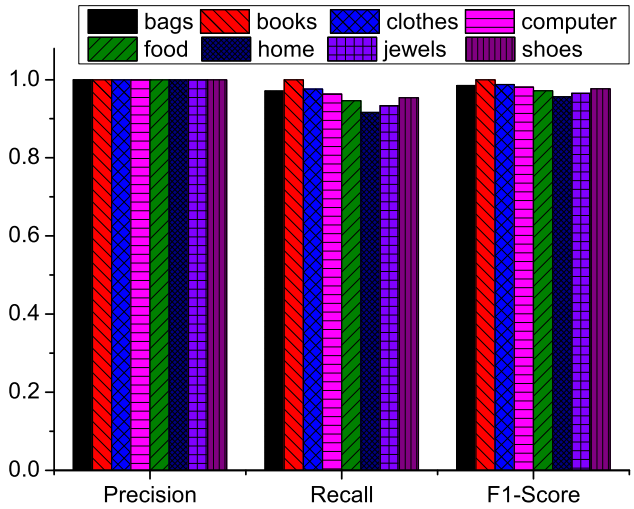

Fig. 13 Extraction performance on http://jd.com

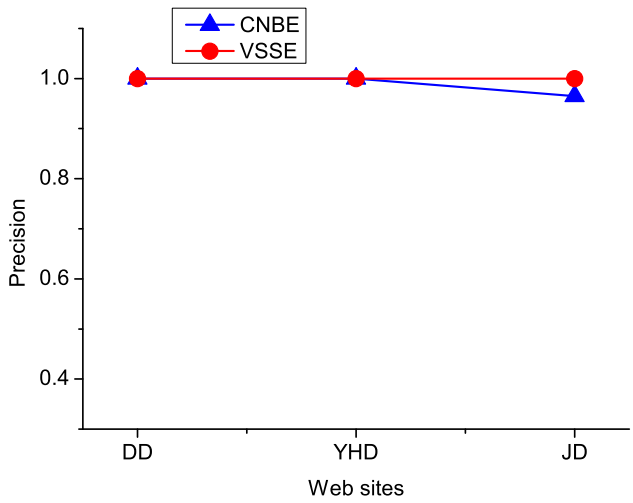

Fig. 14 Precision comparison

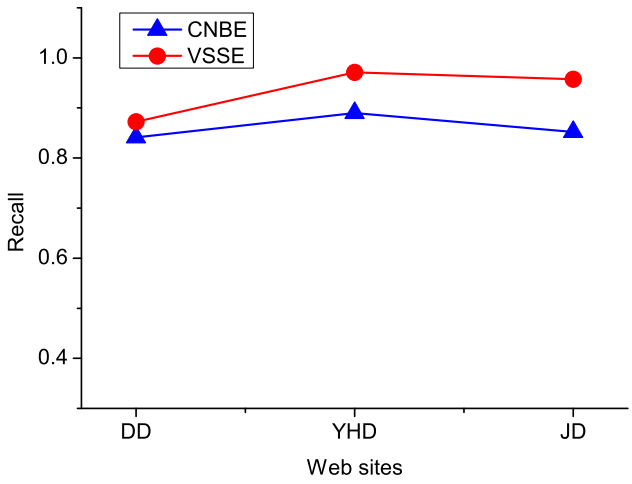

Fig. 15 Recall comparison

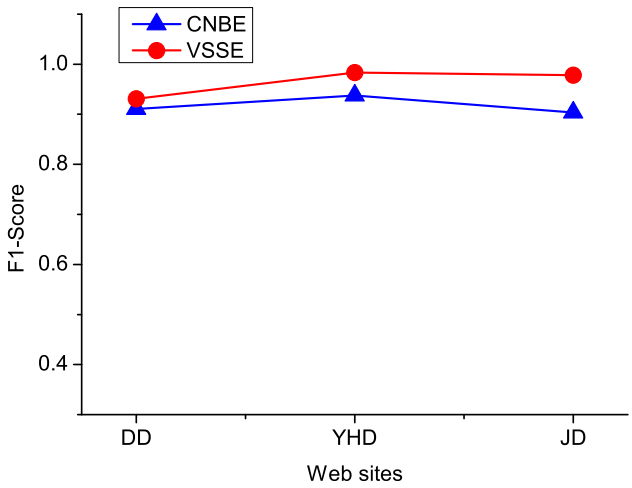

Fig. 16 F1-score comparison

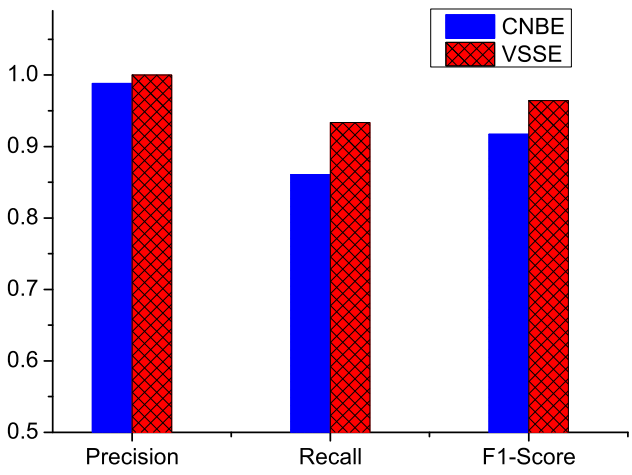

Fig. 17 Overall performance comparison

to ensure both the extraction precision for different kinds of web content and the extraction adaptability on different web environments. Some special extraction characteristics are at the first consideration in the characteristics container, such as visual text information, web page structures, content semantics information, etc. Some other characteristics covering the appearance, structures, etc. can also be inserted into the container and to provide better extraction performance. Web page structures and content semantics similarity are the inherent characteristics of web pages, which can help to identify the extraction areas by aggregating paths or the HMM model and to ensure the steady extraction performance. Especially, the proposed method does not depend on the specific semantics of HTML tags, which ensures the extraction adaptability under different web environments. For the future work, since more editing tags and more nested structures in web pages make a challenge for extraction performance, web page partition and new visual text characteristics should be considered for further performance improvement. On the other side, the extraction characteristics have been an important tradeoff factor between precision and recall, an extensible and auto-filter characteristics container should be designed.

Acknowledgements This work is funded by the National Natural Science Foundation of China (Nos. 61462017, 61363005, U1501252, U1711263, 61662015), Guangxi Natural Science Foundation of China (No. 2017GXNSFAA198035).

References

1. Gupta S, Kaiser G, Neistadt D, Grimm P (2003) Dom-based content extraction of HTML documents. In: Proceedings of the 12th international conference on World Wide Web (WWW '03), pp 207–214
2. Gupta S, Kaiser GE, Grimm P, Chiang MF, Starren J (2005) Automating content extraction of HTML documents. World Wide Web 8(2):179–224
3. Zhang J, Zhang C, Qian W, Zhou A (2011) Automatic extraction rules generation based on xpath pattern learning. In: Proceedings of the 2010 international conference on web information systems engineering (WISS'10), pp 58–69
4. Alam H, Rahman AFR, Hartono R (2001) Content extraction from html documents. In: Proceedings of 1st international workshop on web document analysis (WDA2001)
5. Crescenzi V, Mecca G, Merialdo P (2001) Roadrunner: towards automatic data extraction from large web sites. In: Proceedings of the 27th international conference on very large data bases (VLDB '01), pp 109–118
6. Elmeleegy H, Madhavan J, Halevy A (2011) Harvesting relational tables from lists on the web. VLDB J 20(2):209–226
7. Furche T, Guo J, Maneth S, Schallhart C (2016) Robust and noise resistant wrapper induction. In: Proceedings of the 2016 international conference on management of data (SIGMOD '16), pp 773–784
8. Reis DC, Golgher PB, Silva AS, Laender AF (2004) Automatic web news extraction using tree edit distance. In: Proceedings of the 13th international conference on World Wide Web (WWW '04), pp 502–511
9. Wu G, Li L, Hu X, Wu X (2013) Web news extraction via path ratios. In: Proceedings of the 22nd ACM international conference on information and knowledge management (CIKM '13), pp 2059–2068
10. Wu G, Li L, Li L, Wu X (2016) Web news extraction via tag path feature fusion using ds theory. J Comput Sci Technol 31(4):661–672
11. Qiu D, Barbosa L, Dong XL, Shen Y, Srivastava D (2015) Dexter: large-scale discovery and extraction of product specifications on the web. Proc VLDB Endow 8(13):2194–2205
12. Wu B, Cheng X, Wang Y, Guo Y, Song L (2009) Simultaneous product attribute name and value extraction from web pages. In: Proceedings of the 2009 IEEE/WIC/ACM international joint conference on web intelligence and intelligent agent technology (WI-IAT '09), pp 295–298
13. Cai D, Yu S, Wen J-R, Ma W-Y (2003) Extracting content structure for web pages based on visual representation. In: Proceedings of the 5th Asia-Pacific web conference on web technologies and applications (APWeb'03), pp 406–417
14. Song R, Liu H, Wen J-R, Ma W-Y (2004) Learning block importance models for web pages. In: Proceedings of the 13th International Conference on World Wide Web (WWW '04), pp 203–211
15. Fernandes D, de Moura ES, Ribeiro-Neto B, da Silva AS, Goncalves MA (2007) Computing block importance for searching on web sites. In: Proceedings of the 16th ACM conference on conference on information and knowledge management (CIKM '07), pp 165–174
16. Sun F, Song D, Liao L (2011) Dom based content extraction via text density. In: Proceedings of the 34th international ACM SIGIR conference on research and development in information retrieval (SIGIR '11), pp 245–254
17. Qureshi PAR, Memon N (2012) Hybrid model of content extraction. J Comput Syst Sci 78(4):1248–1257
18. Peters ME, Lecocq D (2013) Content extraction using diverse feature sets. In: Proceedings of the 22nd international conference on World Wide Web (WWW '13 Companion), pp 89–90
19. Ortona S, Orsi G, Buoncristiano M, Furche T (2015) Wadar: joint wrapper and data repair. Proc VLDB Endow 8(12):1996–1999
20. Weninger T, Hsu WH, Han J (2010) CETR: content extraction via tag ratios. In: Proceedings of the 19th international conference on World Wide Web (WWW '10), pp 971–980
21. Uzun E, Agun HV, Yerlikaya T (2013) A hybrid approach for extracting informative content from web pages. Inf Process Manage 49(4):928–944
22. Bing L, Wong T-L, Lam W (2016) Unsupervised extraction of popular product attributes from e-commerce web sites by considering customer reviews. ACM Trans Internet Technol 16(2):12:1–12:17
23. Wong T-L, Lam W, Wong T-S (2008) An unsupervised framework for extracting and normalizing product attributes from multiple web sites. In: Proceedings of the 31st annual international ACM SIGIR conference on research and development in information retrieval (SIGIR '08), pp 35–42

24. Zheng B, Su H, Hua W, Zheng K, Zhou X, Li G (2017) Efficient clue-based route search on road networks. TKDE 29(9):1846–1859
25. Zheng B, Zheng K, Xiao X, Su H, Yin H, Zhou X, Li G (2016) Keyword-aware continuous knn query on road networks. In: Proceedings of ICDE, pp 871–882
26. Zheng B, Yuan NJ, Zheng K, Xie X, Sadiq S, Zhou X (2015) Approximate keyword search in semantic trajectory database. In: Proceedings of ICDE, pp 975–986
27. Zheng K, Zheng B, Xu J, Liu G, Liu A, Li Z (2017) Popularity-aware spatial keyword search on activity trajectories. World Wide Web J 20(4):749–773
28. Zheng K, Su H, Zheng B, Shang S, Xu J, Liu J, Zhou X (2015) Interactive top-k spatial keyword queries. In: Proceedings of ICDE, pp 423–434
29. Weninger T, Palacios R, Crescenzi V, Gottron T, Merialdo P (2016) Web content extraction: a metaanalysis of its past and thoughts on its future. SIGKDD Explor Newsl 17(2):17–23
30. Jsoup. https://jsoup.org/. Accessed 20 Oct 2017
31. Charikar MS (2002) Similarity estimation techniques from rounding algorithms. In: Proceedings of the 34th annual ACM symposium on theory of computing, STOC '02, pp 380–388
32. Zhang H. Nlpir. http://ictclas.nlpir.org/. Accessed 15 Dec 2017