SOFTWARE—PRACTICE AND EXPERIENCE
Softw. Pract. Exper. 0000; **00**:1–22
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/spe

Automatic Review Identification on the Web Using Pattern Recognition

Jeroen van der Meer*and Flavius Frasincar

Erasmus University Rotterdam, Burgemeester Oudlaan 50, 3062 PA Rotterdam, the Netherlands

SUMMARY

In this paper we propose the ARROW (Automatic Review Recognition and annOtation of Webpages) method which is capable of identifying and annotating user submitted reviews on a Web page. The method consists of six steps: data preparation, page-level pattern identification, subjectivity analysis, container structure analysis, review properties identification, and review annotation. For the evaluation we have implemented the method and tested it on various review websites. Based on the performed evaluation we conclude that our method is capable of identifying and annotating the majority of reviews. Copyright © 0000 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: Automatic annotation; user submitted reviews; Google Rich Snippets; RDFa.

1. INTRODUCTION

The Web is an excellent platform for anyone to share his or her visions, ideas, opinions, and much more. An example of such shared information are product reviews. Many websites are dedicated to give anyone the possibility to share his or her thoughts about a certain product. Also, many online stores allow their customers to share their opinion about the stores' products. All these reviews contain valuable information about the products. Finding ways to tap into these information sources and be able to query them would open up realms of possibilities in fields as marketing and e-commerce.

Unfortunately, these review websites do not use a generally accepted structure to publish reviews. At the moment, Web pages containing reviews are formatted using HTML that makes automatic extraction of the review data difficult. The goal of this paper is to ease this process by providing a method that allows for automatic identification and annotation of user submitted reviews from Web pages using an RDFa vocabulary for reviews.

Automatic extraction of user submitted reviews from Web pages is interesting not only from a research perspective, but also from a commercial point of view. In the current days where information is one of the most valuable assets of a company, feedback from customers is vital. This allows a company to know what people like and dislike about its products and how it can improve its products. Reviews on the Web are an excellent source for this kind of feedback. Another application for the automatic extraction and annotation of reviews are decision support systems. These systems aid the user in making decisions, such as which product to buy. However, decision support systems need to import data about the different options to make a decision. Our research can aid companies in identifying and extracting reviews from Web pages.

*Correspondence to: Email: jeroenvdmeer@gmail.com

The innovation of this paper stems from several aspects. This paper is an improvement of [1], our previous work which was the first to propose a method capable of identifying and extracting reviews from the Web. We have improved our previous work by replacing the primitive structure analysis of the Web page with a more advanced structure analysis. Where the structure analysis in our previous paper was only capable of identifying large chunks of text, the structure analysis in this paper is capable of identifying patterns in the HTML markup of Web pages. To our knowledge, this structure analysis combined with the use of subjectivity analysis is a unique approach for automatic review identification on the Web. Furthermore, we target to annotate these reviews using Google's RDFa vocabulary as proposed for Google Rich Snippets. This will allow other applications to make use of the review information, such as search engines like Google and Bing, webshops like Amazon and eBay, and travel agents like Expedia and Booking.com.

Despite the fact that this paper focusses primarily on identifying user submitted reviews on the Web, our approach can easily be extended for identifying other similar data objects. Our method consists of several filters where the input Web pages pass through. By adding or replacing an existing filter with a custom filter, one could make an automatic identifier and annotator for data objects such as weblog comments, forum posts, or questions and answers as on Yahoo! Answers. In that sense, this paper describes a method that can be extended for other purposes also, instead of only identifying and annotating user submitted reviews from Web pages.

This paper consists of seven sections with each section gradually forming the conclusion of this paper. The following section gives an overview of existing research related to our work. Next, we analyze how reviews are represented on the Web and the patterns that can be identified in review websites. Then we discuss the design of our method that is capable of identifying reviews on Web pages. Subsequently, we explain our implementation of this method. We evaluate the performance of our method by analyzing how well our implementation of the method is capable of identifying reviews. Using these evaluation results we form the conclusion to this paper and propose several directions for future research.

2. RELATED WORK

A lot of research has been done on data extraction from the Web. A large proportion of these methods are non-automatic [2, 3, 4, 5, 6, 7, 8]. However, only automatic methods are of interest to allow machines to identify user submitted reviews without human intervention. These automatic methods can furthermore be categorized into three classes: the tag-based methods, the visual cue-based methods, and the text-based methods. Below we discuss each of these method types.

2.1. Tag-based Methods

Tag-based methods use the HTML code of a Web page to transform it into a tag tree. A tag tree is similar to a DOM tree, [9] but only includes element nodes. All other information is discarded. This tree is then traversed while the algorithm of the tag-based method performs a series of actions. The tag-based methods we discuss are Omini [10] and MDR [11].

Omini [10] extracts objects from a Web page in three phases. First, it downloads the Web page associated with the input URL and normalizes the document's syntax. The second phase consists of two steps. In the first step the primary content region is identified in order to weed out sections containing navigation, advertisements, etc. This is done by taking the smallest subtree of the tag tree that contain all the objects of interest. In the last step of the second phase the individual data objects contained by the minimal subtree are separated. This is realized by a combination of five algorithms that identify a sequence of one or more tags that separate the data objects. Once the data objects of interest, together with their boundaries, have been identified in the second phase, the data objects are extracted in the third phase.

MDR (Mining Data Records in Web Pages) [11] extracts data objects from Web pages in three steps. First, the tag tree of the input document is constructed. In the second step the subtrees of the tag tree that contain similar data objects are identified by comparing the tag strings associated with

DOI: 10.1002/spe

each node in the tag tree. These subtrees are called data regions. A tag string consists of the element name of a node, and the names of all the elements in the node subtree in a depth-first traversal. In the last step, the data regions from the previous step are used to search for the data objects. This is done by comparing the tag strings of the child nodes of the data regions. The child nodes that share a similar tag string are labeled as the data objects of interest.

2.2. Visual Cue-based Methods

Visual cue-based methods, as tag-based methods, build a tag tree from the HTML code of a Web page. However, the visual cue-based methods are distinct from the tag-based methods by also using visual cues from the rendered Web page. The type of visual cues that are used differ per method. The visual cue-based methods discussed in this section are ViNT [12], ViPER [13], and MSE [14].

ViNT (Visual information aNd Tag structure based wrapper generator) [12] focuses on extracting information from search result Web pages. Using a set of sample result pages, the process starts by identifying consecutive content line patterns. Content lines are visual lines on the rendered image of the Web page, such as a line of text or a blank line. The content lines that match the same pattern are partitioned into blocks. Subsequently, the boundaries of the individual data objects within these blocks are identified using visual analysis. A set of wrappers are constructed using a tag tree algorithm (similar to MDR) and based on the data objects found after analyzing the visual cues. From these wrappers the most promising is selected and used to extract the data objects. This selection process picks the wrapper that extracts the data region that occupies a large area of the rendered image, is centrally located, contains many characters, and has a large number of data records.

ViPER (Visual Perception-based Extraction of Records) [13] is similar to MDR, but extends the process with an automated visual analysis. First, the input document is scanned for specific patterns to identify the data regions that contain data objects of interest. This is done by comparing the tag strings of all elements that have the same parent node, just like in MDR. Once the data regions have been identified, an analysis of the rendered image of the Web page is performed to identify the individual data objects within the data regions. During this analysis, blank lines (also referred to as "valleys") are used to find the separators between the individual data elements.

MSE (Multiple Section Extraction) [14] is an extension of ViNT. Like ViNT, the primary target of MSE are search result pages. The process starts with the original ViNT method, but is extended with the assumption that there could be multiple data regions on a Web page that content data objects. Next, the dynamic components of the Web page are identified by comparing the different input pages line by line as these also hint towards the locations of the data records. By comparing the result of both processes, the boundaries of the data objects are determined.

2.3. Text-based Methods

Text-based methods differ from the tag-based and visual cue-based methods by not discarding the text nodes on a Web page. These text nodes are thus taken into account when constructing the patterns used for extracting the content of the Web page. In Figure 1 (the right part) words such as "contributions" and "Trip type" are part of the review template. The text-based methods discussed in this section are Roadrunner [15], DeLa [16], and EXALG [17].

Roadrunner [15] uses two sample Web pages to derive a general template that matches both pages. It does so by taking the first page as the wrapper. The second page is used as the sample to compare it against the wrapper. Whenever a tag or a string on the sample does not match with the wrapper, the wrapper is generalized to fit the sample page. Furthermore, Roadrunner is also capable of identifying repetitions when a piece of HTML code has multiple occurrences, and it is capable of identifying optional elements by checking whether a part of the HTML code did occur in a similar part.

Another method, DeLa [16], starts with removing all the uninteresting data, such has page headers, menus, page footers, etc. After this process only the data-rich sections are obtained which could potentially contain data records (such as reviews). Last, DeLa discovers repeating patterns on

DOI: 10.1002/spe

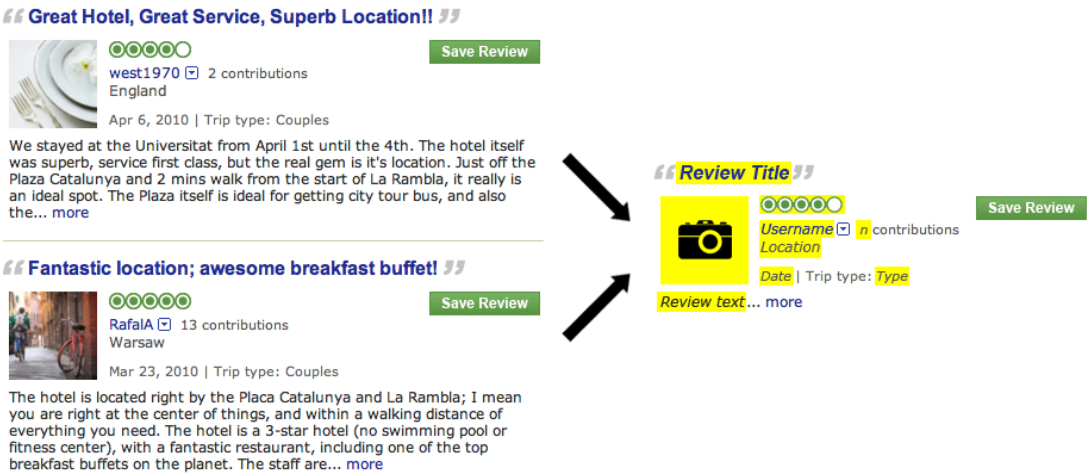


Figure 1. An example of a review pattern.

the page. It will then use the pattern to find all the data records in the data-rich section that matches this pattern.

EXALG [17] consists of two main phases: (1) equivalent class generation, and (2) analysis. In the first phase EXALG computes sets of tokens (a token is a word or an HTML tag) with the same number of occurrences in all input pages. These sets are called equivalent classes. The equivalent classes that occur (multiple times) on multiple pages are most likely part of the template that is used to construct the page. Therefore, EXALG constructs its wrapper using the equivalent classes that contain the elements with the highest occurrence.

2.4. *Subjectivity Analysis*

User submitted reviews generally contain a lot of subjectivity compared to other types of text, such as news articles or informational pages. Subjective sentences can be defined as sentences that express or describe opinions, evaluations, or emotions [18]. Research on the analysis of opinions, sentiment, and subjectivity has become popular nowadays [19, 20, 21].

Pang proposed a system for classifying subjective and non-subjective documents [19]. This systems starts by labeling the sentences in the document as subjective or objective. The objective sentences are ignored. A standard machine-learning classifier is then applied to the subjective sentences.

Another method for subjectivity analysis is based on a sentiment lexicon. LWD (Light-Weight subjectivity Detection mechanism) is such an example [22]. It is an unsupervised mechanism which checks each sentence for subjective words. A document is then labeled as a review when at least k sentences contain at least n subjective words.

2.5. *Annotation Formats*

Aside from Google Rich Snippets to annotate the identified reviews using RDFa, there are alternative options such as Yahoo! Search's SearchMonkey [23]. Just like Google Rich Snippets, SearchMonkey enhances the search results by displaying metadata of the Web page. However, we have chosen Google Rich Snippets over SearchMonkey because of the wide availability of tools and support. An overview of the Google Rich Snippets vocabulary for reviews is given in Figure 2 (where dotted entities are fictive).

For the annotation of the review data, we have chosen to use RDFa. This is because RDFa is a mature markup language that has proven itself in many different applications. Also the large amount of available tools make RDFa a good solution. There are however also good alternatives. The hReview format, which is part of microformats [24], uses standard HTML features to facilitate

DOI: 10.1002/spe

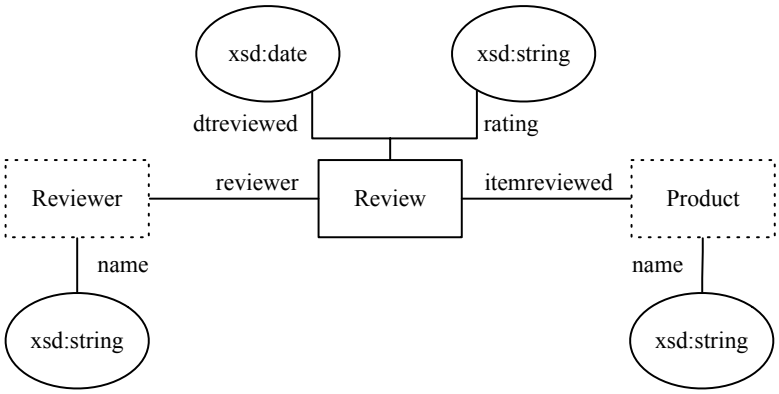


Figure 2. Google Rich Snippets vocabulary for reviews.

the annotations. For example, it uses the `class` attribute to attach a label to an element. Both the RDFa and microformats can be used as their vocabularies are identical.

A second option is Microdata, which is part of the HTML5 specification [25]. Microdata is a simple high level annotation format that extends the HTML language with two new attributes: `itemscope` and `itemprop`. The `itemprop` attribute is used to attach a label to an element that says something about the element labeled with the `itemscope` attribute.

3. ANALYZING REVIEW PATTERNS

With the advent of the Web, and especially Web 2.0, the will of people to share information on the Internet has grown tremendously. This can be seen when analyzing the sheer amount of review websites. These websites are dedicated to allowing users to share information about products. The users of these websites can do so by writing about their experiences with a product, creating an enormous amount of user submitted reviews. To get a better understanding of reviews on the Web, we have analyzed user submitted reviews on a number of different review websites: amazon.com, epinions.com, imdb.com, tripadvisor.com, and yelp.com. This leads us to our first assumption:

Assumption 1 A user submitted review is an opinion and thus highly subjective.

3.1. Patterns on Review Websites

When analyzing a number of different review websites, it quickly becomes clear that the representation of the user submitted reviews share a similar structure. This assumption is vital for constructing a method that is able to identify and annotate the reviews on an arbitrary review website. After all, the knowledge about the structure of the user submitted reviews can be exploited to identify the location of all the reviews on a Web page by determining the pattern they share. In this section we report on the patterns we have found during the analysis of the different review websites, and how these patterns can aid in successfully identifying user submitted reviews on the Web.

Before we dive into the pattern details, we make a distinction between review-level and page-level patterns. When analyzing the review-level patterns, only the reviews are inspected in order to locate the review attributes, such as the name of the author, the date on which the review was posted, and the final rating assigned by the author to the product which was reviewed. When analyzing the page-level patterns, the entire Web page is considered to find the location of each individual review. We start by analyzing the review-level patterns. After understanding the characteristics of an individual review, we consider the page-level patterns.

DOI: 10.1002/spe

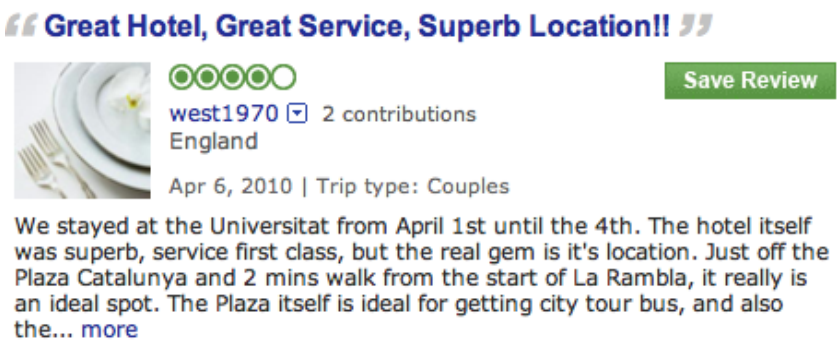


Figure 3. An example of a review.

3.1.1. Review-Level Patterns To be able to identify and annotate the reviews on a Web page, we must first analyze the characteristics of user submitted reviews. Unfortunately the semi-structured nature of Web pages does not make this a simple process, nor does the fact that every review website has a different definition of a review. This definition is often tailored to the reviewed subject. For instance, a review website about hotels might include the trip type and the date of the stay. This data is of course completely irrelevant for reviews about books.

To better understand the characteristics of a review, we have analyzed the Google Rich Snippets RDFa vocabulary we will be using in a later stage to annotate the reviews. This vocabulary defines a review as an object with six properties: the item being reviewed, the rating, the name of the reviewer, the date on which the item was reviewed, the review text, and a summary of the review. An example of a review is given in Figure 3. Here we can clearly see some of the properties defined by the Google Rich Snippets RDFa vocabulary.

Figure 3 is also a good example to show that not all the properties defined in the Google Rich Snippets RDFa vocabulary are always present because this review does not explicitly state the item that is reviewed. So after analyzing the user submitted review websites listed earlier in this section, we have defined our second assumption:

> **Assumption 2** A review is an object containing a subset of the following six properties: item being reviewed, rating, name of the reviewer, date on which the item was reviewed, review text, and a summary of the review, possibly extended with other properties.

In Assumption 2, an object does not necessarily mean an HTML element. The properties can also be scattered across multiple HTML elements. Also, the subset of properties that are present in a review differs for each review website. An example is given in Figure 4. The first review contains ratings for the action factor, special effects, and suspense while the second review does not contain these ratings. However, the differences are often minor. In Figure 4 both reviews use a product rating, reviewer name, review text, summary of the review, and the date of the review. This leads us to our last review-level assumption:

> **Assumption 3** All reviews on a review website share a similar structure.

3.1.2. Page-Level Patterns Now that we know the structure of a review based on Assumption 1, Assumption 2, and Assumption 3, we can locate them on a Web page. Taking into account Assumption 3, we must locate the elements which share a similar structure. When analyzing review websites, it becomes evident that the user submitted reviews are listed one after the other as can be seen in Figure 4, much like user submitted comments on blog posts or news articles.

This structure is also present when analyzing the HTML code of a Web page. Reviews are always "wrapped" inside a review container HTML element. Therefore, the review container is an element which contains elements that have a similar structure. This leads us to our next assumption:

DOI: 10.1002/spe

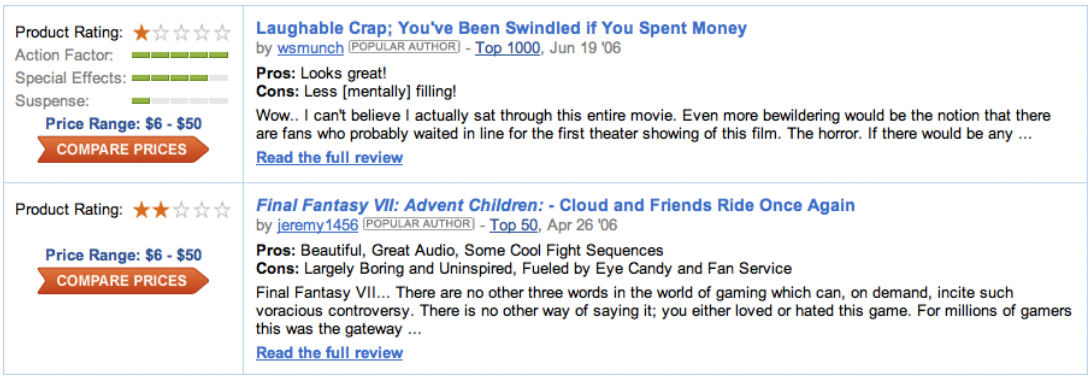
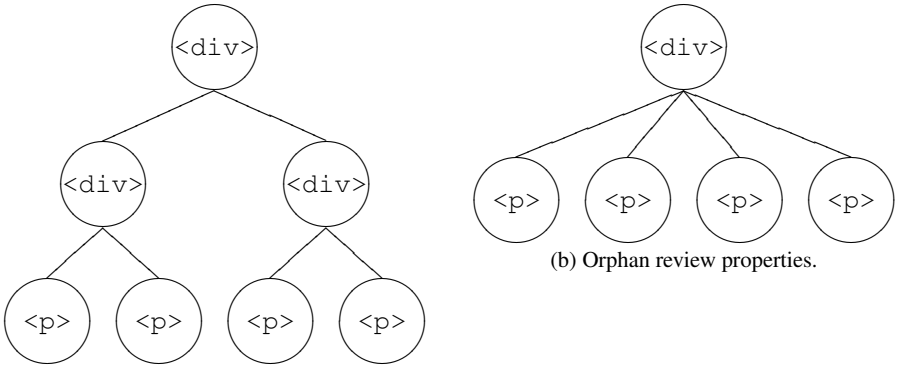


Figure 4. Two reviews on the same website which share a similar subset of review properties.



(a) Group review properties.

(b) Orphan review properties.

Figure 5. Comparison of the two different container element structures.

Assumption 4 A group of reviews is always grouped in one review container.

Despite the fact that each review on a Web page is grouped by one container element, the structure of this container element might differ between two different review websites. There are two possible structures. First, the properties of each individual review are grouped by a grouping element. This is depicted in Figure 5a where the root node represents the container node, the two children nodes of the container node represents the grouping elements, and the <p> elements represents the review properties.

The second possible structure of a container element is depicted in Figure 5b. Like in the first possible container structure, the root node represents the container node. The children nodes of the container node are the review properties which were first grouped together in Figure 5a. Thus, in this case the properties of each individual review are not grouped by an HTML element. These properties share the same parent element: the container element, which leads us to our final assumption:

Assumption 5 The properties of each individual review are embedded in a grouping element, or they are children elements of the review container.

4. ARROW METHOD

In this section we propose Automatic Review Recognition and annOtation of Webpages (ARROW), a method for automatically identifying and annotating reviews from Web pages. This method exploits the assumptions reported in the previous section which allow it to work on many review websites, giving a way to tap into a tremendous amount of review data. This section describes the

DOI: 10.1002/spe

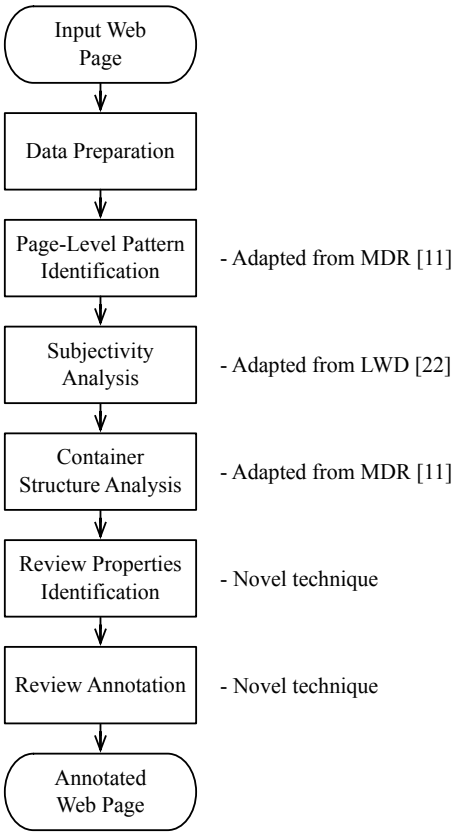


Figure 6. The path of all the processes within the ARROW method.

different steps of the algorithm to identify and annotate user submitted reviews on Web pages. The method is then implemented in a prototype system. This implementation process is described in the next section.

The ARROW method consists of a pipeline with a number of filters. Each of these filters performs a specific algorithm on the input Web page which allows for identification of the reviews and annotation of the identified reviews. An overview of all filters is given in the activity diagram in Figure 6. First, the data is transformed into a usable format. Second, the potential reviews are identified. These potential reviews are then filtered based on an analysis of the subjectivity of the text. For the remaining reviews, the boundaries are identified in the Container Structure Analysis. Between these boundaries, the review properties are searched for. Last, the reviews are annotated.

4.1. Data Preparation

The ARROW method starts by preparing the input data to allow the filters to efficiently work with the Web page. As can be seen in Figure 6, the first step of the method is accepting a valid URL from the user. Subsequently, the HTML markup of the Web page to which the input URL refers to is fetched as a string of characters. Because a string of HTML code is very challenging to use by the filters, the HTML code is transformed into a DOM tree. This DOM tree is an easy to use data structure which allows for easy traversal along the nodes, tree modifications, and selection of specific elements.

DOI: 10.1002/spe

```
<div>
   <ul>
      <li><span>The Netherlands</span></li>
      <li><span>Holland</span></li>
   </ul>
</div>
```

Figure 7. An HTML snippet containing a list with two list items.

4.2. Page-Level Pattern Identification

The second filter of the ARROW method is the Page-Level Pattern Identification. This filter utilizes Assumption 3 and Assumption 4 to locate the potential review containers. A potential review container is an HTML element that contains two or more child elements. These child elements are also HTML elements of which two or more share an identical structure.

To aid this process of identifying the elements with an almost identical structure, an adaptation of the MDR [11] algorithm is used. For each element in the DOM tree a tag string is constructed. A tag string is a serialized form of the element's structure. To create a tag string for an element, first we use the element's tag name, followed by the tag names of all the descendant nodes. For example, take the HTML snippet in Figure 7. The tag string of the `<div>` element is constructed by traversing the element structure in preorder: "`div ul li span li span`".

After the tag string has been calculated for each element in the DOM tree, the tree is again traversed to search for the elements that share a similar structure. Considering Assumption 4 we know that if a Web page contains reviews, these reviews are always grouped by a container element. So during the second tree traversal, each element that contains two or more HTML elements is considered to be a potential review container. If two or more of the potential review container's element have an identical tag string, they are marked as potential reviews.

4.3. Subjectivity Analysis

When all the potential reviews on the input Web page have been found, we use Assumption 1 to distinguish the reviews from the non-reviews. As is stated in Assumption 1, reviews are subjective and thus contain a lot of subjective words such as "good" and "bad". We used an improved version of the LWD system [22]. Unlike the original system, ambiguous subjective words are also included in our lexicon. By checking the textual content of the potential reviews against the lexicon of the LWD system containing words that hint towards subjectivity, the non-subjective elements are filtered out from the set of potential reviews. One must note that this filter performs best when the potential reviews contain at least about 100 words. Very short texts generally have little subjective information.

4.4. Container Structure Analysis

When all the definite review containers are known, their structure will be analyzed to find out whether or not the review properties of each individual review are encapsulated by a grouping element. If not, an artificial grouping element will be added to the DOM tree to group each review in a grouping element. The artificial element is needed because when annotating the reviews, the boundaries of each review must be clear. So we need to know where a review starts and where it ends. By grouping all the review properties of each review into a grouping element we can achieve this. Another reason to analyze the structure of the review containers is to identify the reviews that have been falsely marked as non-reviews (the false negatives) by either the Page-Level Pattern Identification or the Subjectivity Analysis.

For each review container the tag strings of its child elements are used to analyze the structure of the review container. The tag string of each child element of a review container is aggregated in the order in which the elements appear in the DOM tree into one text string. This text string is thus a sequence of tag strings in which a pattern can be searched. This pattern can be used to select

DOI: 10.1002/spe

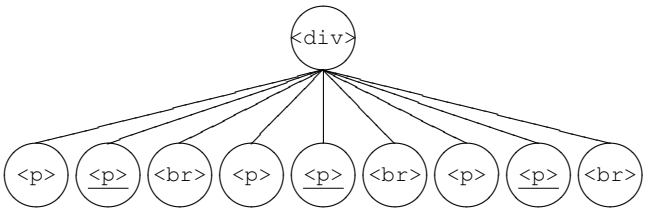


Figure 8. An example of a review container in which the review properties of each review (the <p> elements) are not grouped by a grouping element.

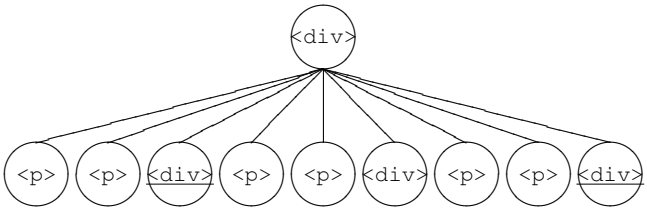


Figure 9. An example of a review container in which one review was falsely marked as a non-review.

which child elements belong together and thus should be grouped by a grouping element, and for identifying the reviews which have been falsely marked as non-reviews.

An example of a review container in which the review properties of each review are not grouped by a grouping element is giving in Figure 8. The root node of the tree represents the review container, the <p> elements represent the review properties, and the
 elements represent line break. Note that the
 elements were ignored during the Page-Level Pattern Identification but are taken into account in the Container Structure Analysis. The <p> elements that have been underlined in the tree have been marked as definite reviews during the Subjectivity Analysis. The tree is traversed in preorder.

As described earlier in this section, in the Container Structure Analysis the tag strings of each child element of the review container are added to a sequence of tag strings. These tag strings are arranged in the same order as the one in which the child elements appear in the DOM tree. In this simple example, this sequence of tag strings looks as follows: "p p br p p br p p br". Most humans can, especially in this very simple example, quickly identify a pattern in this sequence of tag strings, namely: "p p br", which possibly groups the elements of a review.

To create the grouping elements, the boundaries of the reviews must be found. In other words, it must be known where a review starts and where it ends. To do so, we inspect the elements surrounding the elements that have been marked as definite reviews using the tag strings. In Figure 8 these are the underlined <p> elements. As can be seen in the tree, each review element is preceded be an element with the tag string "p" and followed by an element with the tag string "br". The goal is to find the longest possible sequence of elements preceding a review and the longest possible sequence of elements after a review that matches the most reviews.

This method can also be used to identify the reviews that have falsely been marked as non-reviews. Having identified what elements precede a review and which elements are after a review, these sequence of elements can also be used to locate the falsely marked non-reviews. Take for instance Figure 9. In this example the first and third elements have been identified as a review, while the second element has not been identified as a review. However, there is a very high chance this past classification was false, and that the second element is in fact a review.

The falsely marked non-reviews which are not preceded and followed by the same sequence of elements as the definite reviews can also be marked as a definite review if the Levenshtein distance between its tag string and that of any of the definite reviews is above the threshold of 0.3. This threshold was derived as the most optimal threshold by MDR which also used it to compare two tag strings.

DOI: 10.1002/spe

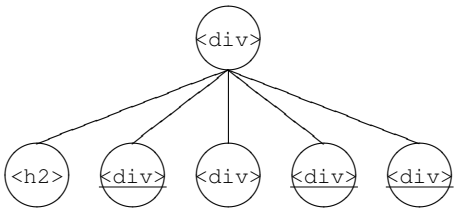


Figure 10. An example of a review container where no pattern can be found in its structure.

Assumption 3 states that all the reviews share a similar structure. So if a review has been marked as a non-review in the Page-Level Pattern Identification, this would be because the review's structure is an exception compared to the structure of the other reviews. If the review was marked as a non-review in the Subjectivity Analysis, this would be because the characteristics of the review text does pass the Subjectivity Analysis. The text could either be too short, badly written, or use slang.

There is also always the possibility that no pattern can be found in the structure of a review container. Take for instance Figure 10. Here the review properties are grouped by `<div>` elements. These are, together with the heading element `<h2>`, grouped by the review container. However, unlike the other `<div>` elements, the second `<div>` element has not been marked as a review. Because the reviews in this example are not preceded or followed by a sequence of elements, the second `<div>` element can not be deduced as a review like in previous examples. Therefore, again based on Assumption 3, we analyze the tag string of the `<div>` elements. If the Levenshtein distance between the non-review's tag string and any of the review elements is above the threshold of 0.3, this non-review element will also be marked as a definite review.

4.5. Review Properties Identification

For the reviews found on the input Web page, the review properties must be identified. From the review, we identify the author, the date, the product that is reviewed, and the rating.

4.5.1. Author To identify the name of the review author, we make use of both keywords and the HTML structure. Many review websites add a specific keyword, such as "By:" or "Author:", in front of the name of the reviewer to emphasize that the text followed by the keyword is in fact the name of the reviewer. These keywords can thus be of great help to identify the location of the reviewer's name.

Another method to identify the location of the author's name is by utilizing the HTML structure of the Web page. Many websites use the `class` attribute to assign classes to an element. Because the HTML5 specification encourages Web authors to use values that describe the nature of the element's content, these `class` attributes can also help for identifying the review author's name.

Many review websites require their users to register at their website. After registration, the user is given an account on the website with their own public profile page. This profile page is then linked to in every review the user writes on the website. This makes identifying the name of the reviewer a lot easier, as we now know that the links in each review are potentially the name of the reviewer. Using this method together with the two methods previously mentioned we can identify the name of the reviewer.

First, we collect all the links inside a review element. For each link, the word preceding the link is checked whether it matches any of the text keywords in Table I. If so, the link text is labeled as the author. If such a link does not exist, the analyzed elements inside the review element are checked. When an element has a `class` attribute that contains any of the class keywords in Table I, the element is further analyzed. If such an element contains an `<a>` element (a link), the link text is labeled as the name of the author. Otherwise, the textual content of the element is labeled as the name of the author.

4.5.2. Date There are many different formats to represent a date. These formats are usually bounded to a specific culture. Despite the existence of standard notations and the globalization of

DOI: 10.1002/spe

Table I. Keywords used to identify the name of the author.

Text keywords	author, by, username, user, reviewer.
Class keywords	author, username, user, reviewer.

Table II. The different date notations and their respective regular expression patterns.

Notation	Regular expression
dd/mm/yyyy	`(\d{1,2})[-/.,\s](\d{1,2})[-/.,\s](\d{2,4})`
yyyy-mm-dd	`(\d{2,4})[-/.,\s](\d{1,2})[-/.,\s](\d{1,2})`
dd MM yyyy	`(\d{1,2})(th)?((\s)of)?\s(\w)\s(\d{2,4})`
MM dd yyyy	`(\w)\s(\d{1,2})(th|,)?\s(\d{2,4})`

```
<title>Red Door Cafe - Pacific Heights - San Francisco, CA</title>
<h1>Red Door Cafe</h1>
```

Figure 11. An example title and h1 element.

the world, many different formats are still used throughout the Web. For instance, one could write a date as “March 11th 2010”, or “11-03-2010”. Both notations look very different, yet represent the same date.

In order to successfully identify the dates of the reviews, we have to consider the many different date formats. These formats are listed in Table II together with the corresponding regular expressions which allow for the identification of the respective date format. In these notations, *dd* stands for the day of the month (1 to 31), *mm* stands for the month (1 to 12), *MM* stands for the name of the month (January to December), and *yyyy* stands for the year.

There are also cases where the month is represented in its ordinal form, or when there is extra punctuation. Both cases can be seen in following notation: “April 18th, 2010”. Also, to separate the day, month, and the year from each other, the comma or full stop characters may be used. These exceptions are also integrated into the patterns listed in Table II. These patterns will be used to match against the review text in order to identify the dates. Once a date has been identified, it will be translated into ISO 8601 date format as required by the Google Rich Snippets RDFa review vocabulary.

4.5.3. Product To identify the product name, we can not use the review elements which were identified in the previous steps of the method. This is because the name of the product that is reviewed is hardly ever mentioned in the review elements. Also, a reviewer might have named a number of related products in order to compare them with the product that is reviewed. In this case, it would very difficult to identify the correct product name from the review text.

In order to identify the product name, the entire Web page is considered. More specifically we investigate the title of the Web page (the `<title>` element) and the top page header (the `<h1>` element). In most cases these two elements contain very valuable information about the subject of the Web page. In the case of review pages, this is usually the name of the product. In general, the Web page title contains a lot of data, such as the name of the website. Due to this, the top header can be of use to filter out the data that is not about the content of the Web page.

Figure 11 displays an example `<title>` element together with an `<h1>` element from the same Web page. After analyzing both elements, it can be concluded that the textual content of both elements share the text “Red Door Cafe”, which is the name of the restaurant that is reviewed on the Web page. This text is used as the name of the product. In case no overlap between the textual content of the `<title>` and `<h1>` elements exists, the text of the `<h1>` element is used as the product name. If there is no `<h1>` element, the `<title>` element is used. Because the `<title>` element is a required element for each Web page, this element is always present.

Prepared using speauth.cls
DOI: 10.1002/spe

Table III. The different rating notations and their respective regular expression patterns.

Notation	Regular expression
4.5 out of 5	`([0-9.,]+)\s(out ?)of\s([0-9.,]+)`
4.5/5	`([0-9.,]+)\s?/\s?([0-9.,]+)`
4 stars	`([0-9.,]+) stars`

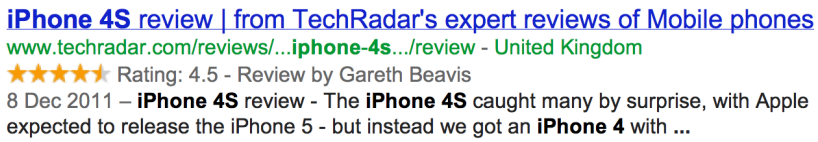


Figure 12. An example of a search result on Google.

4.5.4. Rating Reviews are often concluded with a rating. The rating is often a real number within a certain range. Popular scales are "1 till 5", "1 till 10", and "1 till 100". Alternatively, other symbols such as letters can also be used, just like the American grading system used in schools that goes from A to F. Another option is the use of descriptors such as "excellent", "great", "satisfactory", "bad", and "rubbish". However, we have focused on the ratings that use a number notation instead of letters and descriptors because that is what most websites use (90% of the websites use this notation).

Like identifying the date, pattern extraction can be used to identify and extract the rating of a review. Table III lists the different possible notations with their respective regular expression patterns.

As listed in Table III another format is supported: star images. The Google Rich Snippets also use this notation to represent the rating of the review, as depicted in Figure 12 where four out of five stars are colored. The star images are inserted in the Web page using one or more `<img>` elements which sometimes contain an `alt` attribute which stores the textual fallback content in case the element can not be displayed. The value of the `alt` attribute can thus be used to match against the patterns in Table III in order to extract the ratings encoded in the star images.

4.6. Review Annotation

When the review and its attributes are identified they will be annotated using the RDFa vocabulary designed by Google for its Rich Snippets. For each review, the element that groups all the review properties is considered. This can be either the grouping element created in the Review Container Analysis, or the grouping element that already was present in the original DOM tree. This review element will be annotated as being an instance of the class `Review`. This is done by adding a `typeof` attribute to the review element.

For each review, the review properties which have been identified in the Review Properties Identification are annotated as properties of the review. This is done by adding a `property` attribute to the element containing the property value.

5. ARROW SYSTEM

We have chosen to implement our method as a Web application. Web applications are in general available for everybody, they are platform independent, and they do not require an installation. As a result, this allows anyone to quickly and easily use our method implementation.

The application, which is available at `http://www2.arrow-project.com/`, and has been written in Java. We have chosen Java in order to allow it to be able to run on Google App Engine, and because of its wide variety of libraries. Despite the fact that our application has been written for Google App Engine, Google's free cloud computing platform, the method has been implemented in such a way it can be easily reused in any other Java application. A screenshot of our Web application with a sample result page is shown in Figure 13.

DOI: 10.1002/spe

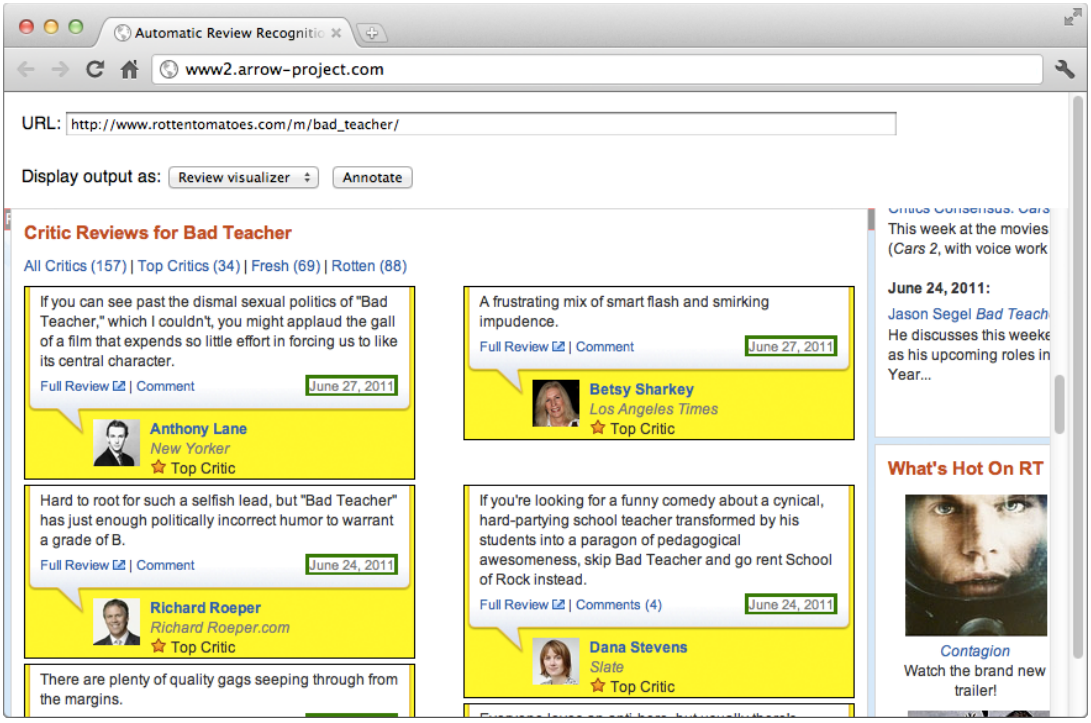


Figure 13. A screenshot of our Web application.

As can be seen in Figure 13, the ARROW application is also capable of visualizing the RDFa annotation results by showing graphically where the reviews on the input Web page are located together with their properties. This allows for easy evaluation of the method results. However, the default output method is of course HTML code of the input Web page with the RDFa annotations added to the review elements and their review properties.

This section covers the implementation of each of the method filters. Like in the previous section, we start with the Data Preparation, followed by the Page-Level Pattern Identification, Subjectivity Analysis, Container Structure Analysis, Review Properties Identification, and the Review Annotation.

5.1. Data Preparation

The first filter of the ARROW method is the Data Preparation. Here the input URL is accepted and parsed using Java's `java.net.URL` class. Once the URL is accepted, a connection is created. Otherwise, if the URL is invalid or the system is unable to open a connection with the URL, the application displays an error message. Using the opened connection, the input stream is fed to W3C's jTidy library [26]. This library is able to transform a stream of characters into a DOM tree, considering the resource to which the URL refers to contain an HTML document. If jTidy is unable to parse the document, the system displays an error message. After this, the DOM tree is pushed to the Page-Level Pattern Identification filter. In our implementation of the ARROW method, JavaScript is outside the scope of this paper. By building an ARROW system with a more sophisticated interpreter that is capable of executing JavaScript, the results might vary from what is presented in this section.

5.2. Page-Level Pattern Identification

By taking the DOM tree given by the Data Preparation, the Page-Level Pattern Identification tries to identify certain structural patterns as described in the previous section. It starts by taking the `<body>` element of the DOM tree and ignoring the `<head>` element. This is because the `<head>`

DOI: 10.1002/spe

Table IV. Test results of review/non-review classification.

Method	Accuracy	Precision	Recall	Specificity	F1-score
lwd(1,4)+	0,53	0,51	0,97	0,08	0,67
lwd(64.3%,2)+	**0,67**	0,73	0,53	0,80	0,61
lwd(5.3%,5)+	0,66	0,61	0,91	0,41	**0,73**
lwd(1,4)	0,55	0,53	0,90	0,20	0,61
lwd(50.1%,2)	**0,67**	0,75	0,50	0,83	0,60
lwd(25.9%,2)	0,60	0,56	0,95	0,25	**0,70**

element represent a collection of metadata for the Web page. Since the reviews are always present in the main content of the Web page, only the `<body>` element is considered in this case.

The Page-Level Pattern Identification then continues with a recursive algorithm. It starts by taking the `<body>` and computes the tag string for each child element as described in MDR. However, some elements have more impact on the structure of a Web page than other elements. For instance the `<em>` element, which is used to stress the emphasis of its textual content, only adds very subtle information about a very small part of the document. A `<div>` element on the other hand could mark the beginning of a new section of the Web page. Therefore the `<em>` element, together with all other text-level elements, are ignored when constructing the tag string.

The full list of elements that are ignored include `<a>`, `<em>`, `<strong>`, `<small>`, `<cite>`, `<q>`, `<dfn>`, `<abbr>`, `<time>`, `<code>`, `<var>`, `<samp>`, `<kbd>`, `<sub>`, `<sup>`, `<i>`, `<b>`, `<mark>`, `<ruby>`, `<rt>`, `<rp>`, `<bdo>`, `<span>`, `<br>`, and `<wbr>`. These elements are the text-level features of HTML5. The list of excluded elements is completed by the obsolete text-level features of HTML5: `<acronym>`, `<rb>`, `<basefont>`, `<big>`, `<blink>`, `<font>`, `<nobr>`, `<s>`, `<spacer>`, `<strike>`, `<tt>`, and `<u>`.

Each element in the DOM tree is considered to be a potential review container when it contains two or more HTML elements. Therefore, we take all the elements that qualify as a potential review container and compare the tag strings of its children elements. When the tag strings of two child elements match, both elements are marked as potential reviews.

5.3. Subjectivity Analysis

The Subjectivity Analysis continues where the Page-Level Pattern Identification stopped. It takes the potential reviews and checks every individual word against the extensive lexicon of English subjectivity words taken from the LWD-system. For each potential review the text contained by the review element is considered. This text is then split into different strings of characters for each sentence. Each word of each sentence is then checked against the LWD-lexicon.

The original method classifies a document as a review when it contains at least one sentence with four subjective words. However, the size of the document is not taken into account. We experimented with a set of 100 reviews and 100 non-reviews. These reviews are taken from different sources and have a different length. Using the original classification method, a lot of false positives were generated.

We have computed new thresholds to improve upon the LWD system. We investigate what percentage of sentences in a document need to be subjective to mark a document as a review, and how many subjective words should a sentence contain to be marked as subjective. We considered all possible subjective sentence percentages between 0% and 100% (with a step size of 0.1%) with at least 2 to 9 subjective words (with a step size of 1). As presented in Table IV, the highest F1-score is obtained when classifying documents as reviews if 5.3% of the sentences contain five or more subjective words. The methods denoted with "+" included ambiguous words.

Using this knowledge, we can classify the potential reviews as reviews or non-reviews. For each of the potential reviews, we take the textual content of the element and check whether at least 5,3% of the sentences have five or more subjective words in them. When the textual content of an element does obey this constraint, it is marked as a definite review and the parent element of the definite review is marked as a definite review container.

DOI: 10.1002/spe

The Subjectivity Analysis proposed in this paper uses an English sentiment lexicon (`http://www.cs.pitt.edu/mpqa/`). This implies that our method in its current form only supports English review websites. However, our method can be easily adapted to different languages by replacing the English sentiment lexicon with a sentiment lexicon of another language.

5.4. Container Structure Analysis

With the definite reviews found in the Subjectivity Analysis, we have a good indication where the reviews are located on the Web page. However, we continue our search by considering the definite reviews and their review container element. The structure of the review containers are further analyzed with the use of the tag strings of the review container's child nodes. The goal of the Container Structure Analysis is to locate the review properties that belong together and the reviews that were missed by the Page-Level Pattern Identification or the Subjectivity Analysis by analyzing the patterns in the review containers' structure.

We start by taking the tag strings of all the child nodes of the review containers that have at least one definite review and arrange them after each other. The order in which the child elements of the review container appear in the DOM tree must be maintained in the sequence of tag strings. In order to simplify the pattern identification of the review container, each distinct tag string is encoded by a letter. This will result in a tag string sequence such as "`ABCABCABC`" instead of the actual tag strings, which can become very large. Although different reviews can have a different tag string, these should both be represented in the tag string sequence as a review, no matter how their tag string looks like. To do so, we replace all the letters in the tag string sequence that represent a review by an exclamation mark. A sample tag string sequence can thus look like "`AB!AB!AB!`". Note that only the letters in the tag string sequence that represent an element that is classified as subjective in the Subjectivity Analysis are allowed to be replaced by an exclamation mark.

The tag string sequence will be used to find the sequence of elements that generally precede or follow a review element. Starting with the first exclamation mark in the tag string sequence, it determines all elements before it. Taking the tag string sequence "`AB!AB!AB!`" as an example, "`AB`" is what appears before the first review. It does the same for all other reviews, but instead it stops when it sees an exclamation mark. It then continues to compare each letter of all the letter sequences one by one from right to left and takes the longest sequence of letters that precede the most reviews. The same is done for the sequence of letters that follow a review.

The review properties can be grouped using the longest letter sequences that precede or follow the most reviews. In the review container a `<div>` element is inserted before each occurrence of the sequence of elements that precede most reviews. These elements, plus all elements after this sequence are appended into the newly created `<div>` element. This is done until a sequence of elements occur that usually follow a review. These elements are appended to the `<div>` element, and after these elements the `<div>` element is closed. Not only does this group all the review properties together into one review element, it also finds the elements that previously were missed by either the Page-Level Pattern Identification or the Subjectivity Analysis.

5.5. Review Properties Identification and Annotation

For each review element identified in the previous steps, the review properties will be identified as described in the ARROW method. Each review property is encapsulated by a `<span>` element. To this `<span>` element a `property` attribute is added. The different possible values for the `properties` attribute depends on the type of review property the `<span>` element is encapsulating. The values are given in Table V.

6. EVALUATION

In order to validate the ARROW method, we have done a series of tests on our Web application. We have constructed a test corpus of 50 Web pages from five different review websites than the ones considered during the design of the ARROW method, namely $ciao.co.uk$, $iTunes$,

DOI: 10.1002/spe

Table V. The values for the `properties` attribute for each review property type.

Property	Description
itemreviewed	The item being reviewed.
rating	A numerical quality rating for the item (for example, 4).
reviewer	The author of the review.
dtreviewed	The date that the item was reviewed in ISO 8601 date format.

Table VI. The test and evaluation results for review identification

(a) Test results for review identification.

Website	TP	TN	FP	FN
ciao.co.uk	142	15926	12	0
iTunes	24	14661	2	6
moviereviews.com	208	13658	0	0
reviewcentre.com	102	14074	3	5
rottentomatoes.com	160	29484	26	95

(b) The review identification evaluation results.

Acc.	Prec.	Spec.	Rec.	F1
0.99	0.92	0.99	1.00	0.96
0.99	0.92	0.99	0.80	0.86
1.00	1.00	1.00	1.00	1.00
0.99	0.97	0.99	0.95	0.96
0.99	0.86	0.99	0.63	0.73

$moviereviews.com$, $reviewcentre.com$, and $rottentomatoes.com$. These Web pages contain at least one user submitted review. The evaluation is done for recognizing the reviews on the Web pages, and recognizing the review properties inside the reviews.

6.1. Review Identification

The evaluation results for recognizing the reviews is presented in Table VIa. This table shows for each website the true positives (TP), the true negatives (TN), the false positives (FP), and the false negatives (FN). In terms of reviews, the true positives are the number of successfully recognized reviews. The true negatives are the HTML elements successfully not considered as reviews. The false positives are the HTML elements that were falsely recognized as reviews. The false negatives are the reviews that were not recognized.

When analyzing the results in Table VIa, it can be concluded that our method is able to identify the majority of reviews on the 50 selected evaluation Web pages. The two exceptions are *ciao.co.uk*, with its relatively high number of false positives, and *rottentomatoes.com*, with its high number of false negatives. The number of false positives can be explained by the fact that the system sometimes labeled a part of a review as a completely new review. This resulted in a true positive review occasionally containing two or even three false positive reviews. The relatively large amount of false negatives on *rottentomatoes.com* can be explained by the fact that this website only displays a small part (the first one or two sentences) of the entire review text. Because of this short length, the Subjectivity Analysis has very little text to analyze, which results in a high number of reviews being classified as false negative. To better understand the data presented in Table VIa, for each website the accuracy, precision, specificity, recall, and F1-score of the review recognition has been calculated. This data is presented in Table VIb. The F1-scores, which consider both precision and recall, indicate a good performance.

6.2. Review Property Identification

After identifying the reviews, the review properties are extracted. In this section we evaluate the identification of the review author, date, rating and the product name similar to how we have evaluated the review identification. First we gather the test data with the number of true positives, these are the review attributes that have been successfully identified by the system. Second, we calculate the true negatives, which are the HTML elements on the Web page that have successfully not been marked as a review property. Third, we calculate the false positives, which are the HTML

Prepared using speauth.cls
DOI: 10.1002/spe

Table VII. The test results of the review properties.

(a) Author identification results.

Website	TP	TN	FP	FN
ciao.co.uk	0	15979	0	142
iTunes	24	14664	0	6
moviereviews.com	208	13660	0	0
reviewcentre.com	56	14040	39	52
rottentomatoes.com	0	29510	0	255

(b) Date identification results.

TP	TN	FP	FN
8	15954	22	137
–	–	–	–
0	23948	0	208
0	86155	0	110
160	29510	0	95

(c) Rating identification results.

Website	TP	TN	FP	FN
ciao.co.uk	1	15978	1	141
iTunes	–	–	–	–
moviereviews.com	1	13654	6	207
reviewcentre.com	96	14085	0	6
rottentomatoes.com	0	29510	0	255

(d) Product identification results.

TP	TN	FP	FN
142	0	0	0
30	0	0	0
110	23948	0	208
0	86155	208	208
0	29510	0	255

Table VIII. The evaluation results of the review properties.

(a) Author identification evaluation.

Website	Acc.	Prec.	Spec.	Rec.	F1
ciao.co.uk	0.99	0.00	1.00	0.00	0.00
iTunes	0.99	1.00	1.00	0.8	0.88
moviereviews.com	1.00	1.00	1.00	1.00	1.00
reviewcentre.com	0.99	0.58	0.99	0.52	0.55
rottentomatoes.com	0.99	0.00	1.00	0.00	0.00

(b) Date identification evaluation.

Acc.	Prec.	Spec.	Rec.	F1
0.99	0.27	0.99	0.06	0.09
–	–	–	–	–
0.99	0.00	1.00	0.00	0.00
0.99	0.00	1.00	0.00	0.00
0.99	1.00	1.00	0.63	0.77

(c) Rating identification evaluation.

Website	Acc.	Prec.	Spec.	Rec.	F1
ciao.co.uk	0.99	0.5	0.99	0.01	0.14
iTunes	–	–	–	–	–
moviereviews.com	0.98	0.14	0.99	0.00	0.01
reviewcentre.com	0.99	1.00	1.00	0.95	0.97
rottentomatoes.com	0.99	0.00	1.00	0.00	0.00

(d) Product identification evaluation.

Acc.	Prec.	Spec.	Rec.	F1
1.00	1.00	1.00	1.00	1.00
1.00	1.00	1.00	1.00	1.00
1.00	1.00	1.00	1.00	1.00
1.00	0.00	1.00	0.00	0.00
1.00	0.00	1.00	0.00	0.00

elements on the Web page that do not contain a review property, but are falsely marked as being a review property. Last, the false negatives, which are the HTML elements that are falsely marked as not containing a review property.

Tables VIIa, VIIb, VIIc, and VIId present the test data of respectively the author identification, date identification, rating identification, and product name identification. In the evaluation of the review identification all HTML elements were considered to be potential reviews. After the test results we calculate the accuracy, precision, specificity, recall, and F1-score. These results are presented in Tables VIIIa, VIIIb, VIIIc, and VIIId. Note that the reviews on iTunes did not contain a review date nor a rating.

DOI: 10.1002/spe

When analyzing Tables VIIIa, VIIIb, VIIIc, and VIIId, we can clearly see a similarity between these tables and Table VIb. The results of the review property identification are strongly connected to those of the review identification. If a review is marked as a false negative by the Page-Level Identification or the Subjectivity Analysis, its properties are also never found because the method does not search for review properties in elements that have not been marked as reviews. Therefore, a false negative review always results in a false negative for its author, date, and rating.

The author identification returned mixed results. While the authors were (almost) perfectly identified on *iTunes* and *moviereviews.com*, they were also well picked out by the system on *reviewcentre.com*. On *ciao.co.uk* and *rottentomatoes.com* however, the system was not able to find any of the review authors. When evaluating the review dates, *ciao.co.uk* and *rottentomatoes.com* actually perform better than the other websites, but still far from perfectly. The sole reason that *ciao.co.uk* was able to identify only eight dates was because on one of the Web pages of this website that was tested was structured differently than the other Web pages. In this structure the system was able to identify the review dates, where it was unable to on the other Web pages.

The identification of the ratings also proved to be challenging. Where on *reviewcentre.com* the ratings were identified almost perfectly, the ratings on the other websites were encoded in such a way our system was unable to identify them. For instance on *ciao.co.uk*, the rating was encoded as an image, but the `<img>` element did not contain an `alt` attribute. Only the rating of one review was correctly identified, because the reviewer also mentioned his or her verdict in the review text. On *moviereviews.com*, the rating was also displayed as an image, but the HTML code did not contain an `<img>` element. The HTML code actually contained five `<div>` elements, one for each of the five starts. Using CSS, the images of the stars were displayed on screen, but our system does not process the CSS code, so it was unable to identify the stars. *rottentomatoes.com* supports a different rating system than what we have discussed in this paper. On this website, a reviewer can rate a movie as fresh (good) or rotten (bad).

The product name identification worked well for *ciao.co.uk*, *iTunes*, and *moviereviews.com*. The main reason why it performs so differently compared to the identification of the other review properties is because the identification process is also very different, as explained in the ARROW Method section. Therefore, even if no review is found, the product name might still be found. Because the same product name is used for all the reviews on a Web page, if the product name for one review is correct, it is correct for all other reviews as well. The product names on *reviewcentre.com* and *rottentomatoes.com* were not successfully identified, because the identified product name contained information such as the name of the website, as the product name is derived using the `<title>` element which usually includes this kind of data. Our method was unable to filter out this data.

6.3. Comparison with other methods

To put the review evaluation results of ARROW in perspective, we have downloaded an MDR [11] implementation[†], which is discussed in the Related Work section, and used it to extract reviews according to the MDR algorithm. There exists no method like ARROW, except for our previous work [1], which focuses specifically on identifying reviews, so we have chosen for a general data object identifier. Additionally, MDR is, like ARROW, an automatic tag-based method. We have used the same set of evaluation Web pages as input for the evaluation process as we did to evaluate ARROW. The MDR results are displayed in Table IXa.

When comparing the MDR evaluation results against the ARROW evaluation results in Table VIb, it becomes evident that ARROW outperforms MDR on all five input websites when we focus on the F1-scores. MDR extracts all data objects that match a specific pattern. In general, this includes navigational menu items and such, which results in a much higher false positives count for MDR. ARROW on the other hand is built specifically to identify the reviews on a Web page, and therefore ignores all other data objects that share a pattern but fail at the Subjectivity Analysis. This results in

[†]http://www.cs.uic.edu/ liub/WebDataExtraction/MDR-download.html

Prepared using speauth.cls
DOI: 10.1002/spe

Table IX. The evaluation results of MDR and our previous work.

(a) The MDR evaluation results of the review identification.

Website	Acc.	Prec.	Spec.	Rec.	F1
ciao.co.uk	0.95	0.13	0.95	0.94	0.23
iTunes	0.97	0.01	0.98	0.13	0.02
moviereviews.com	0.96	0.00	0.98	0.00	0.00
reviewcentre.com	0.99	0.35	0.99	0.16	0.22
rottentomatoes.com	0.98	0.35	0.98	1.00	0.52

(b) The [1] evaluation results of the review identification.

Acc.	Prec.	Spec.	Rec.	F1
0.99	0.89	0.99	0.84	0.87
0.99	0.79	0.99	0.77	0.78
0.99	0.96	0.99	0.64	0.77
0.99	0.96	0.99	0.65	0.79
0.99	0.88	0.99	0.34	0.49

a much higher precision and a lot less noise, which in turn makes ARROW a much better method for identifying and extracting reviews on the Web than a general data object identifier like MDR.

As stated in the Introduction of this paper, the method proposed in the ARROW method section is an improvement based on our previous work [1]. The main improvement in the ARROW method lies within the structure analysis, which analyses the HTML markup of the input Web page. This filter was not present earlier. The improvements can be quantified by evaluating our previous work implementation[‡] using the same set of evaluation Web pages as input for the evaluation process as we did to evaluate ARROW and MDR. The results are displayed in Table IXb.

When comparing our previous work's review identification results against the ARROW evaluation results in Table VIb, we can conclude that ARROW is far superior than our previous work. The Structure Analysis is a much more reliable method for identifying candidate reviews than the Hotspot Identifier is. The Hotspot Identifier was only capable of identifying the large chunks of text on a Web page, which works well up to a certain degree, as can be seen from Table VIb.

7. CONCLUSION

The Web has become an incredible source of information. However, tapping into this information source is still far from trivial. The Semantic Web hopes to overcome these struggles by allowing for better structuring of data on the Web. Subsequently, this allows for great opportunities, such as Google Rich Snippets which gives more appealing and relevant information in Google's search results. However, the Semantic Web has yet to become mainstream. To help Web authors make their website ready for the Semantic Web, we propose the ARROW method. This method is capable of automatically identifying and annotating reviews on Web pages using the Google Rich Snippets vocabulary. Also, the proposed method could be implemented on a reverse proxy server that annotates reviews on a Web server with their corresponding metadata.

We have analyzed the user submitted reviews on a number of different review websites and came to the conclusion that there are definitely similarities between the reviews on the different review websites. This led us to make five assumptions. First, a review is per definition an opinion and thus highly subjective. Second, a review is an object containing a subset of the following six properties: product name, rating, reviewer, date, review text, review summary, and possibly extended with other properties. Third, all reviews on a review website share a similar structure. Fourth, a set of reviews is always grouped in one review container. Last, the properties of each individual review is grouped by a grouping element, or they are not, and in this case are children elements of the review container.

Based on our assumptions, we presented the ARROW method. This method consists of six steps: Data Preparation, Page-Level Pattern Identification, Subjectivity Analysis, Container Structure Analysis, Review Properties Identification, and Review Annotation. Subsequently, we

[‡]http://www.arrow-project.com/

DOI: 10.1002/spe

have presented an implementation of the ARROW method. This application uses the pattern recognition algorithm proposed in the method which allows any Web page that contains user submitted reviews to be annotated for the Google Rich Snippets.

Reflecting back on the research presented in this paper, we can state that using pattern identification the review regions on a Web page can be recognized. User submitted reviews generally share a similar template throughout the entire review website. Recognizing this template thus allows for identifying most of the reviews on the Web pages. This heuristic is implemented in the ARROW method. The evaluation of this method shows the usefulness of pattern identification for automatically identifying and annotating user submitted reviews.

The method scores well on identifying and annotating reviews. In our evaluation, the method achieved F1-scores between 0.73 and 1.00 for the identification of reviews. For identifying the review author, F1-scores between 0.55 and 1.00 were obtained, except for on two websites where our method was unable to identify the review authors. The F1-scores for the review date identification ranged between 0.00 and 0.77. For the review rating, F1-scores were between 0.00 and 0.97. Last, the F1-scores for the product name identification ranged between 0.00 and 1.00. These scores also indicate the superiority of ARROW over ARROW 1 and MDR, which only achieved F1-scores between respectively 0.49 and 0.87, and between 0.00 and 0.52, respectively.

7.1. Future Work Directions

Utilizing the knowledge about structural patterns in the HTML code of a Web page has proven to be a successful method for automatic identification and annotation of user submitted reviews. However, the evaluation has also shown that the method proposed leaves room for improvements.

Our method fails to identify the reviews that contain short review texts. This was illustrated by the case of *rottentomatoes.com* in the evaluation. This calls for a method that is more flexible and robust. In our method an element is considered a review when at least 5.3 percent of the sentences in its textual content contains five or more subjective words. This does not take into account the size of the text. A method that scales these thresholds depending on the length of the text could significantly improve upon our current method. Additionally, the usage of a JavaScript interpreter would create better results, given the increasing usage of this scripting language on the Web. Also, we would like to extend our date recognition to consider also textual references to time (e.g., today, yesterday, last month, etc.), in addition to the numeric ones.

A second possible future work direction for review identification would be the use of an entirely different perspective: the human perspective. Websites are currently made to be read by humans. This is the main root of the problem why it is difficult for computers to extract the relevant data from Web pages. However, instead of using the HTML code of a Web page, which can easily be processed by a computer, it could also be possible to use the rendered image of the Web page, as generated by Web browsers.

Using the rendered image of a Web page, the Web page can be analyzed like humans would read the Web page. As stated earlier, Web pages are primarily made for humans to read, so taking this perspective to analyze a Web page would make a lot of sense. Unfortunately methods for optical analysis of Web pages are still far from mature, which would make this approach very challenging. On the other hand, it opens the door to new dimensions and unseen opportunities for identifying and analyzing user submitted reviews.

Using the visual information from a Web page for data gathering is not something new. ViPER [13] and MSE [14] also make use of the visual cues of a Web page, but no methods exist that uses these visual cues to extract review data from Web pages. A simple example can be the possibilities for product name identification. As described in the ARROW Method section, the product name is derived from the `<h1>` and `<title>` elements on a Web page. This simple method works on most websites, but fails to work on some others, despite that all websites we used in our research had the product name encoded in a text header. When taking the visual cues into account, a new method that identifies the headers based on their visual characteristics, such as the text size, can be used instead. We think this method would be able to return the product name with a much higher reliability.

Prepared using speauth.cls
DOI: 10.1002/spe

REFERENCES

1. Van der Meer J, Boon F, Hogenboom F, Frasincar F, Kaymak U. A Framework for Automatic Annotation of Web Pages Using the Google Rich Snippets Vocabulary. *2011 ACM Symposium on Applied Computing (SAC 2011)*, ACM, 2011; 765–772.
2. Hammer J, McHugh J, Garcia-Molina H. Semistructured Data: The TSIMMIS Experience. *1st East-European Symposium on Advances in Databases and Information Systems (ADBIS 1997)*, Springer, 1997; 1–8.
3. Crescenzi V, Mecca G. Grammars Have Exceptions. *Information Systems* 1998; **23**(8):539–565.
4. Adelberg B. NoDoSE - A Tool for Semi-Automatically Extracting Structured and Semistructred Data from Text Documents. *ACM SIGMOD International Conference on Management of Data (SIGMOD 1998)*, ACM, 1998; 283–294.
5. Hsu CN, Dung MT. Generating Finite-State Transducers for Semi-Structured Data Extraction from the Web. *Information Systems* 1998; **23**(8):521–538.
6. Califf ME, Mooney RJ. Relational Learning of Pattern-Match Rules for Information Extraction. *16th Conference on Artificial Intelligence (AAAI 1999)*, American Association for Artificial Intelligence, 1999; 328–334.
7. Muslea I, Minton S, Knoblock C. A Hierarchical Approach to Wrapper Induction. *3rd Annual Conference on Autonomous Agents (AGENTS 1999)*, ACM, 1999; 190–197.
8. Laender A, Ribeiro-Neto B, Da Silva A. Debye – data extraction by example. *Data & Knowledge Engineering* 2002; **40**(2):121–154.
9. W3C. Document Object Model (DOM) Level 3 Core Specification. `http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/` 2004.
10. Buttler D, Liu L, Pu C. A Fully Automated Object Extraction System for the World Wide Web. *21st International Conference on Distributed Computing Systems (ICDCS 2001)*, IEEE Computer Society, 2001; 361–370.
11. Liu B, Grossman R, Zhai Y. Mining Data Records in Web Pages. *9th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD 2003)*, ACM, 2003; 601–606.
12. Zhao H, Meng W, Wu Z, Raghavan V, Yu C. Fully Automatic Wrapper Generation for Search Engines. *14th International Conference on World Wide Web (WWW 2005)*, ACM, 2005; 66–75.
13. Simon K, Lausen G. ViPER: Augmenting Automatic Information Extraction with Visual Perceptions. *14th ACM International Conference on Information and Knowledge Management (CIKM 2005)*, ACM, 2005; 381–388.
14. Zhao H, Meng W, Yu C. Automatic Extraction of Dynamic Record Sections From Search Engine Result Pages. *32nd International Conference on Very Large Data Bases (VLDB 2006)*, VLDB Endowment, 2006; 989–1000.
15. Crescenzi V, Mecca G, Merialdo P. RoadRunner: Towards Automatic Data Extraction from Large Web Sites. *27th International Conference on Very Large Data Bases (VLDB 2001)*, Morgan Kaufmann, 2001; 109–118.
16. Wang J, Lochovsky FH. Data Extraction and Label Assignment for Web Databases. *12th International Conference on World Wide Web (WWW 2003)*, ACM, 2003; 187–196.
17. Arasu A, Garcia-Molina H. Extracting Structured Data from Web Pages. *ACM SIGMOD International Conference on Management of Data 2003 (SIGMOD 2003)*, ACM, 2003; 337–348.
18. Wiebe J, Wilson T, Bruce R, Bell M, Martin M. Learning Subjective Language. *Computational Linguistics* 2004; **30**(3):277–308.
19. Pang B, Lee L. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. *42nd Annual Meeting on Association for Computational Linguistics (ACL 2004)*, Association for Computational Linguistics, 2004; 271.
20. Wilson T, Hoffmann P, Somasundaran S, Kessler J, Wiebe J, Choi Y, Cardie C, Riloff E, Patwardhan S. OpinionFinder: A System for Subjectivity Analysis. *Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing 2005 (HTL/EMNLP 2005)*, Association for Computational Linguistics, 2005; 34–35.
21. Boiy E, Hens P, Deschacht K, Moens MF. Automatic Sentiment Analysis in On-line Text. *Conference on Electronic Publishing 2007 (ELPUB 2007)*, 2007; 349–360.
22. Barbosa L, Kumar R, Pang B, Tomkins A. For a Few Dollars Less: Identifying Review Pages Sans Human Labels. *2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2009)*, Association for Computational Linguistics, 2009; 494–502.
23. Yahoo! Search. SearchMonkey - YDN. `http://developer.yahoo.com/searchmonkey/` 2011.
24. Çelik T. hReview 0.3 Microformats Wiki. `http://microformats.org/wiki/hreview` 2010.
25. Hickson I. HTML5, a Vocabulary and Associated APIs for HTML and XHTML. `http://www.w3.org/TR/html5/` 2011.
26. Quick A. About JTidy. `http://jtidy.sourceforge.net/` 2011.

DOI: 10.1002/spe