A Framework for Automatic Annotation of Web Pages Using the Google Rich Snippets Vocabulary

Jeroen van der Meer
jeroenvdmeer@gmail.com

Ferry Boon
ferry.boon@gmail.com

Frederik Hogenboom
fhogenboom@ese.eur.nl

Flavius Frasincar
frasincar@ese.eur.nl

Uzay Kaymak
kaymak@ese.eur.nl

Econometric Institute
Erasmus University Rotterdam
PO Box 1738, NL-3000 DR
Rotterdam, the Netherlands

ABSTRACT

One of the latest developments for the Semantic Web is Google Rich Snippets, a service that uses Web page annotations for displaying search results in a visually appealing manner. In this paper we propose the Automatic Review Recognition and annOtation of Web pages (ARROW) framework, which is able to identify reviews on Web pages and to annotate them using RDFa attributes. The ARROW framework consists of four steps: hotspot identification, subjectivity analysis, information extraction, and page annotation. We evaluate an implementation of the framework by using various Web sites. Based on the evaluation we conclude that our framework is able to properly identify the majority of reviews, reviewed items, and review dates.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*linguistic processing*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*text analysis*; I.7.m [**Artificial Intelligence**]: Document and Text Processing—*miscellaneous*

General Terms

Languages, design, management

Keywords

Annotation, information extraction, Google Rich Snippets, RDFa

1. INTRODUCTION

The World Wide Web is home to an ever increasing and already vast amount of information. A large portion of this information is available in such a way that it is easy for humans to comprehend it. For machines on the other hand it is almost impossible to understand information contained in the Web pages. One of the pillars of the Semantic Web is to define the content of the Web pages semantically (i.e., as concepts with meaning) in order to make data machine understandable. The ability of computers to automatically process and interpret data will support new functionality on the Web.

One of the latest functionalities added to the Web is Google's Rich Snippets. This is a service for Web page owners to add semantics to their (existing) Web page using the semantic vocabulary provided by Google. Up until now the vocabulary is rather limited in its number of concepts (Person, Review, Review Aggregate, Product, and Organization). Yahoo! has a similar service provided by SearchMonkey which uses a collection of publicly available vocabularies.

When a Web site has annotated its pages with Google's vocabulary, Google's search engine will highlight the recognized concepts and show a brief summary of the Web page and the concepts' attributes using visual cues. An example of a review aggregate in Google Rich Snippets is shown in Fig. 1. This figure depicts the search result of a sample Web page that contains reviews about a resort & spa which are annotated with Google's vocabulary. Google has extracted the annotated metadata about the reviews from this Web page to display it in its search results. Such a search result will stand out in a long uniform list of search results and it is considered to be more likely that people will click on that link.

Rich Snippets are only a small possibility of what annotated Web pages can do in the future. When sites are annotated it is just a small step to more sophisticated and advanced search capabilities. For example, when searching

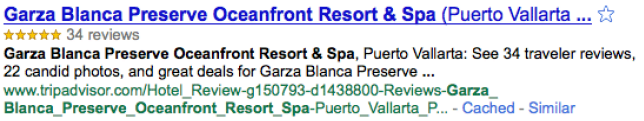

Figure 1: An example of a *Review Aggregate* Rich Snippet in Google search results.

SAC'11 March 21-25, 2011, TaiChung, Taiwan.

for the company "Philip Morris", with Rich Snippets one is able to state that all results with "Philip Morris" as a person should be ignored. Another possibility would be to let Google search for a product within a certain price range with at least a certain percentage of positive reviews.

If a Web site is built from structured data from a database, annotating the Web pages is fairly straightforward. It would be sufficient to identify its concepts in the generated pages and add the attributes to the Web page while generating the HTML output. Not all Web pages are built from databases and thus pre-generation of annotations is not always possible. In these cases the Web pages must be annotated manually, which can be a very time consuming and tedious job. In this paper, we discuss a method to automatically read and annotate Web pages, using the necessary RDFa attributes as defined in Google Rich Snippet's vocabulary. An excerpt of the RDFa used in the example in Fig. 1 is given in Fig. 2.

```
<div xmlns:v="http://rdf.data-vocabulary.org/#"
     typeof="v:Review">
   <span property="v:itemreviewed">
      Garza Blanca Preserve Oceanfront Resort & Spa
   </span>
   <span property="v:reviewer">
      Jane Doe
   </span>
   <span property="v:rating">
      5 stars
   </span>
   <span property="v:dtreviewed">
      21th March 2011
   </span>
   <p property="v:summary">
      This is an absolutely beautiful hotel for the money.
      Lovely property, architecture and landscaping.
   </p>
</div>
```

Figure 2: A Google Rich Snippets annotation of a Review entity, using the RDFa format.

In this paper, we propose the Automatic Review Recognition and annOtation of Web pages (ARROW) framework, which reads Web pages, identifies reviews, and annotates the pages with the RDFa attributes defined by Google Rich Snippets. Using algorithms based on heuristics, subjectivity analysis, and entity recognition techniques, we implement the framework in a tool that takes a URL as an input, and outputs a Google Rich Snippets-compliant review page.

The remainder of this paper is organized as follows. First, we will discuss related work in Sect. 2. Next, we introduce the ARROW framework in Sect. 3, followed by details on its implementation in Sect. 4. Subsequently, Sect. 5 presents a performance evaluation on our algorithms. Finally, we draw conclusions and discuss some directions for future work in Sect. 6.

2. RELATED WORK

With the vast amounts of customer reviews available on the Web, a great amount of research already has been done on taking advantage of these rich information sources. Most of this research is done in the context of opinion mining. Several methods have been proposed for mining, summarizing, and comparing reviews [6, 7, 10]. However, in contrast to our goals, most of these algorithms aim to infer new information from existing reviews, whereas we aim to extend the underlying HTML code by adding semantics after identifying information on the Web page.

Even though a lot of research has already been done on information extraction from Web pages [4], there is little work on extracting reviews from Web pages. These Web information extraction systems work by means of a wrapper to extract the data from Web pages, and allow the user to query this data. In this paper, we focus only on unsupervised systems, as they can be fully automated and do not require pre-annotated documents for training. Based on the content of a Web page, an unsupervised method tries to find a pattern on the Web page. This pattern can be a set of recurring HTML tags or specific text strings.

Examples of unsupervised Web information extraction systems are RoadRunner [5], EXALG [1], DeLa [17], and DEPTA [9]. To identify the attributes of the reviews, e.g., author, date, etc., these systems employ methods such as person or date recognition. These methods can be divided into tag-based approaches, text-based approaches, and hybrid approaches. The tag-based approaches derive a wrapper for the Web site based on the structural characteristics of a Web page. Text-based approaches focus on the textual content of a Web page. Lastly, the hybrid approaches are a combination of the tag-based and text-based approaches and thus contain elements of both methods.

2.1 Tag-Based Approaches

An example of a tag-based approach is EXALG, which consists of two main phases: (1) equivalent class generation, and (2) analysis. In the first phase, EXALG computes sets of tokens with the same number of occurrences in all input pages. These sets are called equivalent classes. For example, in every HTML page the tags `<html>` and `<body>` usually occur only once. These tags are therefore part of the same equivalent class together with all other tokens that appear only once in every page. 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.

2.2 Text-Based Approaches

Natural Language Processing (NLP) can also be used to extract the relevant attributes from Web pages. It might not be always the case that attributes such as rating or reviewer name are clearly annotated using markup languages, but instead are part of the review text. Another application of NLP in our case would be to check whether or not data records extracted by the Web information extraction system are actually reviews. Named Entity Recognition (NER) is a commonly used method for determining proper nouns in text.

Named entity recognition, also known as entity identification or entity extraction, is used for classifying elements in text into predefined categories such as persons, organizations, quantities, time values, etc. There are two main techniques used for NER systems, i.e., linguistic grammar-based techniques and statistical models. Systems relying mostly on the grammar-based techniques obtain a better precision but at the cost of a lower recall and they are more time consuming than the statistical systems. Statistical NER systems on the other hand often need a large amount of manually annotated data as training data. Previous research shows

that even the latest NER systems are quite brittle, meaning that when the system is developed for one domain it typically does not perform well on others [13]. A lot of time and effort is involved in making the system work properly in a certain domain. This is true for both statistical and grammar-based systems.

The latest versions of NER-systems have near-human performance on English texts (on the specific domains they are trained for). These systems, developed for the Message Understanding Conference (MUC7), scored a 93.39% for F1-measure whereas human annotators scored 97.60% and 96.95% [11]. State of the art Java-based NER taggers include the Stanford NER and the Illinois NER.

2.3 Hybrid Approaches

RoadRunner takes two sample Web pages to find structural and textual commonalities between them and subsequently uses these to generate the wrapper. Another method, DeLa, starts with removing all the uninteresting data, such has page headers, menus, page footers, etc. After this process we obtain only the data-rich section which could potentially contain data records (such as reviews). Finally, DeLa looks for continuously repeating patterns on the page. It then uses the pattern to find all the data records in the data-rich section that matches this pattern. The third, and last hybrid method that we investigate is DEPTA, which consists of three steps. First of all, it creates a tag tree of the Web page. This is similar to a Document Object Model (DOM) tree but in the tag tree only tags are considered. Secondly, substrings for all child nodes in the tree that share the same parent are compared. If two substrings are similar to a certain degree, the nodes are labeled as data regions. The last step handles situations where two data regions are not followed right after each other.

2.4 Subjectivity Analysis

One could consider a review to be a written opinion of a person about a certain subject, e.g., a product. Because opinions are subjective, we can safely assume that reviews have a substantially larger amount of subjective words in them than an objective news message or an informational piece of text would have. Subjective sentences can be defined as sentences that express or describe opinions, evaluations, or emotions [18]. Recently the computational treatment of opinions, sentiment and subjectivity have attracted a great deal of attention [12, 3]. It is useful for companies and Web sites to create summaries of people's experiences and opinions extracted from reviews or just the review's overall polarity (i.e, positive or negative). In this way, companies and Web sites can easily access and analyze how people think about certain products or services, without having to read all the individual reviews.

Pang and Lillian [12] propose a system for classifying a document as being a review or non-review. The functionality of their framework is twofold: (1) it labels the sentences in the document as either subjective or objective and discards the objective sentences, and (2) it applies standard machine-learning classifiers (i.e., Naive Bayes and Support Vector Machine-based polarity classifiers) to the subjective sentences to identify their polarity (positive or negative). This system prevents the classifier to consider irrelevant or misleading text by eliminating sentences with no or too little subjectivity.

Previous research efforts emphasize the difficulty of the subjectivity mining task due to the wide variety of ways to express emotions and the fact that a mix of objective and subjective sentences can be used. Lexicon based techniques use the sentiment of single words to determine the sentiment of a sentence or a document. Machine learning techniques on the other hand use features to determine sentiment. Often used features are unigrams and part-of-speech data.

An unsupervised LightWeight subjectivity Detection mechanism (LWD) has been proposed in [2], which can compete with some of the most sophisticated supervised methods. The LWD method checks every sentence for subjectivity words and a certain document is identified as a review when at least k sentences contain a minimum of n subjectivity words.

2.5 Annotation

There are three different methods available to annotate reviews. One of these is Microformats [8]. Microformats is a collection of formats that makes the extraction of semi-structured information such as reviews possible. In the case of reviews, the hReview microformat can be encountered on various Web sites. Secondly, the W3C is working on extending the HTML language, as part of the HTML5 specification, to allow native support for annotations as described by the Microdata format. The third and final option is RDFa. RDFa extends XHTML with a set of attributes that allow the XHTML code to be enriched with metadata. Although RDFa is aimed towards extending XHTML, its attributes can also be used in HTML as most RDFa parsers will recognize these attributes.

Similar to ARROW, the OpenCalais Web Service [16] allows for automatic annotation of textual content. In contrast to the ARROW framework, OpenCalais employs Natural Language Processing (NLP) techniques and machine learning to perform the analysis of the content. However, despite the large set of supported entities in OpenCalais, reviews are not included.

3. ARROW FRAMEWORK

Google Rich Snippets supports a limited vocabulary of RDFa entities and their attributes. An excerpt of this vocabulary is given in Fig. 3.

Our main focus is on recognizing and annotating the review entities and their attributes in Web pages. The proposed ARROW framework for automatically annotating review pages by adding RDFa annotations to a Web page is composed of three stages. Fig. 4 shows the main steps in our framework.

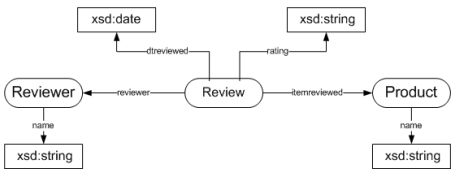

Figure 3: Excerpt of the vocabulary supported by Google Rich Snippets.

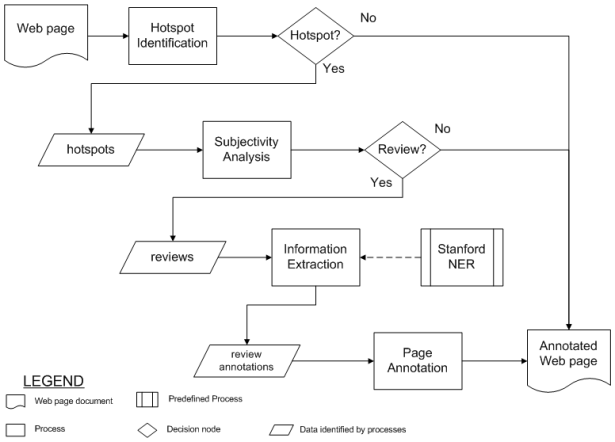

Figure 4: Stages of our recognition algorithm and their intermediate products.

3.1 Hotspot Detection

After normalizing the data, we continue with identifying the potential reviews or *hotspots* of the page. Usually, reviews are characterized by blocks of text, which you find less in some parts of the page, such as the page header, navigation, footer, etc. These parts are usually structured by large amounts of HTML elements. If we want to identify reviews, we thus want to find the elements that contain lots of textual content. Let us consider the example snippet of HTML code depicted in Fig. 5.

```
<h1>
   Intel Core i7-975 Extreme And i7-950
   Processors Reviewed
</h1>

<div>
   <p>
      Page <span class="page-number">1</span> of
      <span class="num-pages">15</span>
   </p>
</div>
```

Figure 5: Two sample `h1` and `div` HTML snippets. The `h1` element contains more textual content than the `div` element.

The `h1` element clearly contains more textual content than the `div` element. In fact, the `h1` element contains a string of 64 text characters ("\n␣␣␣Intel␣Core␣i7-975␣Extreme␣And ␣i7-950\n␣␣␣Processors␣Reviewed\n"), whereas the `div` element only contains 34 text characters. This includes the textual content of the `span` elements, and the spaces and line breaks around the child nodes of the `div` element. The entire `h1` element contains 73 characters, whereas the `div` element consists of 116 characters. Thus, $\frac{64}{73} \times 100\% \approx 88\%$ of the `h1` element is pure text. The `div` element is made only of $\frac{34}{116} \times 100\% \approx 29\%$ text. Using this ratio we can determine how much of an element consists of text. More formally, this text-to-content ratio, the $TTCR$, can be denoted as

$$TTCR = \frac{L_{text}}{L_{DOM}}, \quad (1)$$

where the number of characters in text is denoted by L_{text} and the total number of characters within the DOM tree is represented by L_{DOM}. The elements with a high text-to-content-ratio are labeled as hotspots.

The more textual content an element contains, the higher the $TTCR$ will be. However, when an element has textual content, there are always at least two tags present in a string: the start tag and end tag. Therefore the $TTCR$ will never equal 1, and thus $TTCR < 1$. Now that we can give each element a $TTCR$, we will use a threshold to differentiate between hotspot elements and non-hotspot elements.

3.2 Subjectivity Analysis

After identifying a hotspot, we need to find if indeed it represents a review. A review can be defined as a subjective view on a certain topic, as opposed to an objective view which describes only facts about said topic. To show that reviews are more subjective than non-reviews, a corpus of 100 review pages (containing 1 or more reviews) and 100 non-review pages has been collected from various Web sites. The corpus covers a wide variety of subjects and consists of text differing in size and writing styles. A distribution was made for both sets according to the y percentage of sentences containing x subjectivity words. It is clear from Fig. 6 that reviews contain a significantly bigger percentage of sentences with more subjective words in them as non-reviews.

In order to be able to inspect the hotspots, we use an improved version of the LightWeight subjectivity Detection mechanism (LWD) as proposed by [2]. Initial experiments on a training set of 100 review pages and 100 pages without reviews, from different sources and with different lengths and layouts, show that this method generates a lot of false positives. We improve on their method by taking the length of the review into account. The original method only checks whether a document has at least k sentences which contain at least n subjectivity words. We propose to classify a document as a review when at least m percent of all sentences in the text contains a minimum of n subjectivity words.

3.3 Information Extraction

In order to be able to extract the attributes from reviews, we employ several methods, depending on the characteristics of the attributes.

3.3.1 Author

We use a Named Entity Recognizer (NER) to find persons in the review. We inspect the first and last entity identified by the NER system and retrieve the place of the entity in the DOM tree. The higher in the tree hierarchy, the more likely

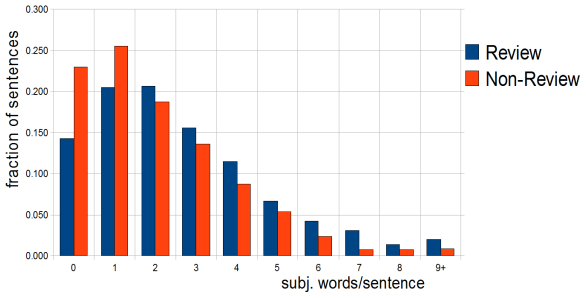

Figure 6: Distribution of sentences with subjectivity words in our test corpus.

Table 1: Date notations and their associated regular expression patterns.

Notation	Regular expression
dd/mm/yyyy & mm/dd/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})`

it is that the entity is important and thus more likely to be the reviewer. In the case of a tie the first entity is chosen because with most reviews the reviewer is introduced at the beginning of a review.

3.3.2 Date

Dates can be represented in many different formats. For instance, one could write a date as "January 15th 2010" or "15-01-2010". Sometimes a date is not limited to just the day of the month, the month, and the year, as is the case in the two example dates mentioned earlier. Often the time is added, which in some cases includes seconds or even milliseconds. Then there is also the difference between the 12-hour clock and the 24-hour clock.

To successfully extract the date of a review, we have to take many different notations into account. The date formats are listed in Table 1, together with a regular expression that can be matched against text to find dates. In these notations, *dd* stands for the day of the month (1 through 31), *mm* represents the month (1 through 12), *MM* is used for the name of the month (January through December), and *yyyy* denotes the year.

Sometimes, the days of the month are represented in their ordinal form or extra punctuation is added. For example, this is the case in 'April 18th, 2010'. Also, instead of the slash sign (U+002F) the comma (U+002C), dash (U+002D), and full stop character (U+002E) can also be used as the separator between the day and the month and between the month and the year. To cope with all these different formats, pattern matching is used to identify these date and time formats in the reviews.

3.3.3 Product

The name of the product that is being reviewed can be located on numerous locations on the Web page. Every Web site maintains their own style. It is often hard to identify the product in the review content. Other related products are also frequently mentioned. In book or movie reviews, for instance, the product is often compared to previous work of the same author or director.

Through empirical observations we discovered that the product name is often mentioned in the `title` and `h1` elements of the HTML code. The `title` element represents the title of the Web page, whereas the `h1` element represents a heading of the highest hierarchical order on the Web page. Thus, it makes sense to analyze these important elements to find the product name on a review page.

Usually the `title` element contains more than just the product name. It is common for the `title` element to also contain the name of the Web site. This is where the `h1` element can be of use. Each `h1` element on the page can be checked against the `title` element for any overlapping tokens. These overlapping tokens are likely to form the product name, e.g., the `title` and `h1` elements depicted in Fig. 7.

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

Figure 7: An example title and h1 element.

In this example, the textual content of the `h1` element matches the first three words in the `title` element. What also can be seen in the `title` element is the use of a dash (U+002D) as a separator. Other popular separators are the slash sign (U+002F) and vertical line (U+007C). These can be used to split the textual content of the title element into tokens. These tokens can be used as a fallback method when comparing the `h1` element with the `title` element results in no match. The first token is then the most likely candidate to represent the product name, as we have found through empirical observations.

3.3.4 Rating

In more than 95% of our analyzed Web pages, we observe that a review is concluded with a final grade or rating. The rating r is often represented by a real number bounded by the rating scale used on the Web site, i.e., $r_{min} \leq r \leq r_{max}$. Popular scales are "1 till 5", "1 till 10", and "1 till 100". In 90% of the cases, images are used for representing ratings. Of these ratings, about 5% use letters (similar to the American grading system used in schools that goes from A to F) and 15% use descriptors, e.g., "excellent", "great", "satisfactory", "bad", "rubbish". However, because most Web sites use an (alternative) number to assign the rating, we have focused on numerical ratings.

Just as is the case with identifying the date of a review, pattern extraction is used to recognize and extract the rating of a review. Table 2 lists the different rating notations ARROW recognizes, together with their respective regular expression patterns.

A popular format to assign the rating is by the use of star images. Such is the case on Google for instance, as depicted in Fig. 1. An advantage of such a format is that it is easy for the user to identify the range of the rating. In Fig. 1,

Table 2: Rating notations and their associated regular expression patterns.

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

Figure 8: A screenshot of our Web application.

four out of the five stars are colored yellow, thus assigning a rating of four out of five to the product. These star images are coded in the Web page's HTML code using one or more `img` elements. These `img` elements sometimes have an `alt` attribute specified which is the textual fallback content in case the element can not be displayed. This `alt` element can thus be checked for patterns described earlier, as images are hard to be comprehended by machines.

3.4 Page Annotation

When the review and its attributes are identified, the framework annotates pages using Google's RDFa vocabulary designed by Google for its Rich Snippets. Annotating involves putting tags around the identified key elements of the review.

4. ARROW IMPLEMENTATION

We have implemented the ARROW framework as a Web application, available at `http://www.arrow-project.com/`. The Web offers a great platform that allows the application to be used on any kind of operating system on a computer anywhere in the world. For implementing the framework, we have used the Java programming language. Together with the Apache Tomcat server, the Java programming language offers excellent methods to develop a Web application according to the framework described in Sect. 3, as it is supported by a wide variety of available libraries. Figure 8 shows a screenshot of our Web application.

Figure 8 shows the two sections of the user interface: (1) the input fields, and (2) the output canvas. A URL can be entered in the input form. The application will then render the Web page in its annotated form or display its hotspots, depending on the output type that has been specified in the input form. Of course, this is just one of the many possibilities to implement the ARROW framework.

The preliminary step of our system is data transformation. In this step the specified URL is retrieved and transformed into a DOM tree. Transforming the HTML or XHTML code into a DOM tree eases the manipulation of the Web page's content compared to having the content as a string of characters. It makes accessing specific tokens easy and because of the popularity of this standardized data structure, a large number of APIs to access it are readily available. In order to transform the HTML or XHTML code of the Web page into a DOM tree we employ W3C's jTidy [15] library. This package offers a very simple, yet effective, API to transform the string of tokens into a DOM tree.

4.1 Hotspot Detection

To calculate the $TTCR$ we traverse the DOM tree. During this process we calculate the L_{text} and L_{DOM} values for each element we encounter. These are then used to calculate the element's $TTCR$ as described in Eq. 1. Whenever the $TTCR \geq 0.85$, the element is marked as a hotspot. The cut-off value of 0.85 has been empirically found through experimenting with values between 0 and 1 with a step size of 0.01 on a training set containing 100 pages with 1 or more reviews, targeting high recall in order not to miss reviews (even if the precision is relatively low), and proved to provide the best results.

A summary of the test results is presented in Table 3. In these experiments, as well as in our other experiments discussed in this paper, we observe the accuracy ACC, precision PRC, recall REC, specificity SPC, and F1-score $F1$.

4.2 Subjectivity Analysis

In our implementation of the ARROW framework, we use an improved version of the LWD system and instead of using only positive and negative opinion words, we include ambiguous subjectivity words as well. The reason is that some of these words can be used as both positive and negative, like "limitless" or "major", or they amplify the meaning of the surrounding subjectivity words, for example the words "absolutely" and "immensely". These words are assigned "neutral" subjectivity in the lexicon. However, we prefer to call them ambiguous to prevent confusion with the term "neutral" which denotes lack of subjectivity.

Table 3: Test results of hotspot identification.

$TTCR$	ACC	PRC	REC	SPC	$F1$
0.25	0.97	0.00	0.00	1.00	0.00
0.50	0.98	0.66	0.66	0.99	0.66
0.75	0.99	0.71	0.78	0.99	0.75
0.80	0.99	0.90	0.80	1.00	0.85
0.85	0.99	0.87	0.87	1.00	0.87
0.90	0.99	0.88	0.49	1.00	0.63
0.95	0.98	0.95	0.35	1.00	0.52
1.00	0.97	0.00	0.00	1.00	0.00

For our improved LWD algorithm, the optimal values of percentage of subjective sentences m and minimum number of subjectivity words per sentence n (i.e., 5.3% and 5, respectively) have been determined empirically, targeting for the highest F1-score. In our experiments on a training set containing 100 pages with 1 or more reviews and 100 pages without reviews, values of m ranged between 0% and 100% with a 0.1% increment, and values of n ranged between 2 and 9 with an increment of 1. A summary of the test results is presented in Table 4, showing that including the ambiguous subjectivity words improves both the accuracy and F1-score. In the table the methods denoted with "+" included "neutral" subjective words.

4.3 Information Extraction

For each of the reviews discovered during the subjectivity analysis we identify the attributes of the reviews. These attributes will then be properly annotated using the RDFa vocabulary for Google Rich Snippets. Even though two high performance NER systems, i.e., the Illinois NER system and the Stanford NER system, obtain high F1-scores on the CoNLL-2003 NER shared tasks (0.9080 and 0.8686, respectively), their scores on the Web tasks are much lower (0.7489 and 0.7250, respectively) [14]. The downside of the Illinois system is however that it uses extensive gazetteers, whereas the Stanford NER system uses conditional random field models, making the whole process significantly slower than the Stanford system. As we implement our framework as a Web application, speed is more important than accuracy (the user will never employ a slow system) and therefore we employ the Stanford NER system.

5. ARROW EVALUATION

This section evaluates the ARROW framework in more detail. Using the optimal cut-off values for text-to-content ratio, minimum percentage of subjective sentences, and minimum number of subjectivity words per sentence determined in Sect. 4, we evaluate the framework on review identification and attribute identification, based on a test set that is different from the training set used for determining optimal parameter values. On average the review annotation process took less than 1 second for each Web page. Figure 1 shows an application using a Web page that was annotated using ARROW. In this example application Google has used the annotated reviews to enhance its search results.

5.1 Review Identification

To assess the review recognition performance, we test the tool on a selection of 100 English review Web pages and 100 non-review Web pages. We annotate the reviews on the Web page manually, then present the URL to the ARROW application. We then compare the manually annotated document with the framework output, and analyze them on true and false positives and negatives, accuracy, precision, recall, specificity, and F1-score. The results are shown in Table 5, with non-review Web sites marked with an asterisk (*).

Table 4: Test results of review/non-review classification.

Method	*ACC*	*PRC*	*REC*	*SPC*	*F*1
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

Table 5: Test results for review identification.

Web site	*ACC*	*PRC*	*REC*	*SPC*	*F*1
`tripadvisor.com`	0.60	0.93	0.53	0.85	0.68
`epinions.com`	0.44	0.98	0.37	0.95	0.53
`imdb.com`	1.00	1.00	1.00	1.00	1.00
`yelp.com`	0.54	1.00	0.53	1.00	0.69
`cnn.com`*	0.74	0.00	0.00	0.74	0.00

These results underline that we obtain good results on precision and specificity, yet varying results on accuracy and recall. The results also show us that the framework works better on some Web sites than on others, caused by type of content, specific Web site structures, etc. For example, the high performance of `imdb.com` can be explained by the structure of the Web site, as these reviews have not too many HTML tags and are very verbose. Furthermore, `cnn.com` is a Web site which does not contain reviews, explaining why reviews were not found by our framework.

5.2 Attribute Identification

We perform a similar experiment in order to assess the performance of review attribute identification. For all true positives, i.e., reviews that are correctly identified by our framework as being reviews, we check for correct recognition of review attributes. As attribute recognition is performed on plain text (the content of the review) we aggregate our results for the different Web sites, as shown in Table 6.

Based on these results, we can conclude that our framework does a good job on finding the item reviewed, date, and rating, but performs poorly on detecting the authors. This can be explained by the ambiguity of the names used on the Internet, as many people use nicknames on the Internet rather than their real (full) names. This makes the automatic identification of people quite hard as the Stanford NER system has been trained to recognize real names. This also explains the high number of false positives, because when the actual reviewer is missed, it often returns a wrong entity as being the reviewer.

6. CONCLUSIONS

Google Rich Snippets is one of the possible improvements that semantic annotations can add to the Web. Google Rich Snippets allows for a more appealing presentation by emphasizing some specific concept properties. Unfortunately, there are not yet many Web sites that support this vocabulary. In

Table 6: Test results for attribute recognition.

Attribute	*ACC*	*PRC*	*REC*	*SPC*	*F*1
reviewer	0.02	0.07	0.03	0.00	0.04
item reviewed	1.00	1.00	1.00	1.00	1.00
date	0.85	0.98	0.86	0.00	0.92
rating	0.49	0.71	0.62	0.00	0.66

order to allow existing Web sites to make use of Google Rich Snippets, we have proposed the ARROW framework in this paper, which aims to automatically identify and annotate reviews on Web pages using the Google's vocabulary.

The framework consists of four steps: hotspot identification, subjectivity analysis, information extraction, and page annotation. We have focused on a subset of Google Rich Snippets, which considers reviews. We have presented an implementation of the framework, which is a Web interface, but it can also be used as a plugin for other Web applications aiming to make use of the provided annotations. Next to the visual annotation, the Web interface provides for the RDFa annotation. Even though the framework makes use of Google Rich Snippets, our approach should be easy to use also for other vocabularies due to the genericity of our components: tag-to-text-ratio, named-entity-recognizer, sentiment-analysis, etc. Subsequently, the innovation of this paper lies in combining techniques from pattern extraction, sentiment analysis, named entity recognition for annotating Web pages using the Google Rich Snippets vocabulary.

Our review identification results underline that we obtain good results on precision and specificity, yet varying results on accuracy and recall. The results also show us that the framework works better on some Web sites than on others, caused by type of content, specific Web site structures, etc. Furthermore, our framework does a good job on finding attributes, like item reviewed, date, and rating, but performs poorly on detecting the authors, which can be explained by the ambiguity of author names used on the Internet.

As future work, we suggest to extend our framework to cover other elements from the Google Rich Snippets vocabulary, e.g., recipes, videos, and organizations. The only constraints it poses to Web pages are the existence of large portions of text intermixed with tags. Of course the parameters have to be adapted accordingly and some components need to be changed (e.g., different tag patterns for recipes and video recognition), replaced (e.g., no sentiment analysis is required to recipes for example), or introduced (e.g., action lists recognition for recipes). In addition, we would like to improve on our framework by being able to extract ratings from images when no caption is given. Also, one could take into consideration that many reviews lack an explicit rating, e.g., a grade or a number of stars. As Google Rich Snippets accepts a rating based on a scale of 1 to 5, it would be useful to investigate ways of calculating ratings based on review texts using, for example, sentiment analysis methods.

7. REFERENCES

[1] A. Arasu and H. Garcia-Molina. Extracting Structured Data from Web Pages. In *2003 ACM SIGMOD International Conference on Management of Data (SIGMOD 2003)*, pages 337–348. ACM, 2003.

[2] L. Barbosa, R. Kumar, B. Pang, and A. Tomkins. For a Few Dollars Less: Identifying Review Pages Sans Human Labels. In *Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2009)*, pages 494–502. ACL, 2009.

[3] E. Boiy, P. Hens, K. Deschacht, and M.-F. Moens. Automatic Sentiment Analysis in On-line Text. In *11th International Conference on Electronic Publishing (ELPUB 2007)*, pages 349–360, 2007.

[4] C.-H. Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan. A Survey of Web Information Extraction Systems. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1411–1428, 2006.

[5] V. Crescenzi, G. Mecca, and P. Merialdo. RoadRunner: Towards Automatic Data Extraction from Large Web Sites. In *27th International Conference on Very Large Data Bases (VLDB 2001)*, pages 109–118. Morgan Kaufmann Publishers Inc., 2001.

[6] M. Hu and B. Liu. Mining and Summarizing Customer Reviews. In *10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2004)*, pages 168–177. ACM, 2004.

[7] M. Hu and B. Liu. Mining Opinion Features in Customer Reviews. In *19th National Conference on Artificial Intelligence (AAAI 2004)*, pages 755–760. AAAI Press / The MIT Press, 2004.

[8] R. Khare and T. Çelik. Microformats: A Pragmatic Path to the Semantic Web. In *15th International World Wide Web Conference (WWW 2006)*, pages 865–866. ACM, 2006.

[9] B. Liu, R. Grossman, and Y. Zhai. Mining Data Records in Web Pages. In *9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2003)*, pages 601–606. ACM, 2003.

[10] B. Liu, M. Hu, and J. Cheng. Opinion Observer: Analyzing and Comparing Opinions on the Web. In *14th International World Wide Web Conference (WWW 2005)*, pages 342–351. ACM, 2005.

[11] A. Mikheev, M. Moens, and C. Grover. Named Entity Recognition without Gazetteers. In *9th Conference of the European Chapter of the Association for Computational Linguistics (ACL 1999)*, pages 1–8. ACL, 1999.

[12] B. Pang and L. Lee. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization based on Minimum Cuts. In *42nd Annual Meeting on Association for Computational Linguistics (ACL 2004)*, pages 271–278. ACL, 2004.

[13] T. Poibeau and D. Dutoit. Generating Extraction Patterns from a Large Semantic Network and an Untagged Corpus. In *SEMANET: Building and Using Semantic Networks Workshop collocated with the 19th International Conference on Computational linguistics (COLING 2002)*, pages 1–7. ACL, 2002.

[14] L. Ratinov and D. Roth. Design Challenges and Misconceptions in Named Entity Recognition. In *13th Conference on Computational Natural Language Learning (CoNLL 2009)*, pages 147–155. ACL, 2009.

[15] SourceForge.net. JTidy. `http://jtidy.sourceforge.net/`, 2010.

[16] Thomson Reuters. OpenCalais. `http://www.opencalais.com/`, 2010.

[17] J. Wang and F. H. Lochovsky. Data Extraction and Label Assignment for Web Databases. In *12th International World Wide Web Conference (WWW 2003)*, pages 187–196. ACM, 2003.

[18] J. Wiebe, T. Wilson, R. Bruce, M. Bell, and M. Martin. Learning Subjective Language. *Computational Linguistics*, 30(3):277–308, 2004.