# Hyperlocal Event Extraction of Future Events

Tobias Arrskog, Peter Exner, Håkan Jonsson, Peter Norlander, and Pierre Nugues

Department of Computer Science, Lund University
Advanced Application Labs, Sony Mobile Communications
{tobias.arrskog,peter.norlander}@gmail.com
hakan.jonsson@sonymobile.com
{peter.exner,pierre.nugues}@cs.lth.se

**Abstract.** From metropolitan areas to tiny villages, there is a wide variety of organizers of cultural, business, entertainment, and social events. These organizers publish such information to an equally wide variety of sources. Every source of published events uses its own document structure and provides different sets of information. This raises significant customization issues. This paper explores the possibilities of extracting future events from a wide range of web sources, to determine if the document structure and content can be exploited for time-efficient hyperlocal event scraping. We report on two experimental knowledge-driven, pattern-based programs that scrape events from web pages using both their content and structure.

## 1 Introduction

There has been considerable work on extracting events from text available from the web; see [1] for a collection of recent works. A variety of techniques have been reported: [2] used successfully data-driven approaches for the extraction of news events while knowledge-driven approaches have been applied to extract biomedical [3], historical [4], or financial events [5] among others.

Much previous research focuses on using the body text of the document, while some authors also use the document structure. For example, [4] apply semantic role labelling to unstructured Wikipedia text while [6] use both the document structure and body text to extract events from the same source.

The focus of this paper is on extracting future events using the body text of web pages as well as their DOM structure when the content has multiple levels of structure. We naturally use the body text from the web page as itthat contains essential information, e.g. time, date, and location instances. We also exploit the DOM structure as a source of information. Although HTML embeds some sort of structure, the actual structure is not homogeneous across websites. We report on the problem of extracting event information from a variety of web pages and we describe two systems we implemented and the results we obtained. .

### 1.1 Properties of Local Events

The events we are interested in are those that typically appear in calendars and listings, such as cultural, entertainment, educational, social, business (exhibitions, conferences), and sport events, that attract athe general and large public may have an interest in.

The end goal of this project is to be able to serve users with information about events that match their current interest and context, e.g. using location-based search, by aggregating these events from hyperlocal sources.

Event aggregators already exist, e.g. *Eventful* and *Upcoming*, that collect and publish event information, but they tend to only gather information about major events in cooperation with organizers or publishers. By contrast, we want to extract existing information directly from the publisher.

The main challenge is time-efficient scaling since there is a great number of hyperlocal organizers and sources as well as variations in the formats and DOM structure of the sources and ambiguity. We may also have to deal with missing, ambiguous, or contradictory information. For example, locations can appear in the title:

> Concert – Bruce Springsteen (This time in the new arena),

and contradict the location indicated elsewhere. Another example is a title:

> Outdoor dining now every Friday and Saturday

containing date information which narrows or sometimes contradicts the dates indicated elsewhere on the page.

FThe domain we are interested deals with future events form. This is a very wide area, where only few historically-annotated data is available. This makes a statistical approach problematic, at least initially. Instead, we chose a knowledge-driven, pattern-based approach, where we process both the structure of HTML documents and their content. We analyzse the content using knowledge of the event domain, e.g. event keywords.

In this paper, we report on the problem of extracting event information from given web pages and we describe two systems we implemented and the results we obtained.

### 1.2 Applications and Requirements for Event Structures

From the possible properties of an event, wWe chose to extract the *title, date, time, location, event reference (source)* and *publisher* which answers the *wWhen, where*, and *what* questions aboutof thean event. These are however the most basic attributes, and for a useful application, further information could be extracted, including topic, organizer, cost and target audience.

We set aside In this paper, we do not cover the semantic representation of event data, but future research may need to address representing the above attributes in existing event data models.

## 2 System Architecture

### 2.1 Designing a Simple Scraper

For each site in the list, we created a unique script. These scripts contained a hand-crafted set of rules to extract the correct information for that specific site. This may require a good deal of manual effort as we naturally have toTo expand the list of additional hand-crafted scripts is required, which leads to high costs when scaling to multiplemany sources..

In order to limit scaling costs, the scripts need to be simplistic. For this reason, we decided to A chosen limit ation was that the internal structure of the information in the events needs to be the same between each other, so that a small set of rules can extract the information from all the events.

### 2.2 Designing a Generic Scraper

We investigated if it would be possible to create a generic scraper which could handle all websites without manual labour.

The first step to generically scrape a website is to find all the pages that contain events. This is currently done using domain knowledge, i.e. the system is given only pages which are known to contain events. The possibilities to find pages without manual labour is further discussed in Sect. 5. The system uses six steps to scrape the events from a given web page. Figure 1 shows the system architecture. We implemented the first three steps using the ad-hoc scripts of Sect. 2.1.

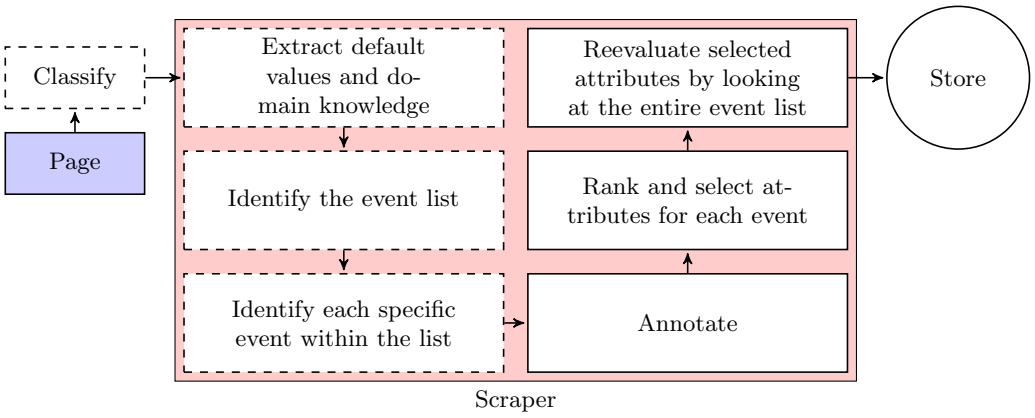


**Fig. 1.** The implemented generic scraper. Dashed boxes use manually written, site-dependent scripts.

### 2.3 Attribute Annotation and Interpretation

The system uses rules to annotate and interpret text. The benefit of a rule-based system is that it can both parse the text and create structured data. As previous work suggests, extracting the time and date of events can be solved through rules. While problematic, the system is able to extract named entities, for example named locations as well. To do this, the system uses three major rules:

1. Keyword detection preceding a named location, e.g looking for *location:* or *arena:*
2. Keyword detection succeeding a named location, for example a city
3. Structured keyword detection preceding a named location. e.g. look for *location* or *arena* when isolated in a separate structure. As an example: ***location*** *Boston* which corresponds to "`<b>location</b> Boston`" using HTML tags.

When the rules above return a named location, we query it against a named location database. Using these rules and a database lookup, we can minimize the false positives.

### 2.4 Attribute Ranking and Selection

The system uses domain knowledge to choose what data to extract:

- The system extracts only *one* title and chooses the most visually distinguished text it can, implied by the DOM structure
- Dates and times are following a hierarchy of complexity, where it takes those of highest complexity first. Some sites used a structure where event structures were grouped by date. To avoid false positives with dates in these event structures, the scraper choose dates between the event structures if less than half of the event structures contained dates.
- The extraction of the location for the event was done in the following order: If the event structure contained a location coordinate, choose it. Otherwise use a default location. If the event site had no default location, use the most commonly referred city in the event structure.

## 3 Evaluation

### 3.1 Scoring

We evaluated the performances of the simple and generic scrapers and we compared them with a scoring defined in Table 1.

**Table 1.** Criteria for full and partial scoring for the test set.

| | Full match |
|---|---|
| Title | Lexicographic distance to correct = 0 |
| Date | Resulting date(s) equal to correct date(s) |
| Time | Resulting start time equals correct start time (minute) |
| Location | Result within 1000 m of correct |

| | Partial match |
|---|---|
| Title | Result contains correct title |
| Date | Full match or if result contains at least one of correct date(s) |
| Time | Full match or if result contains at least one of correct start time(s) |
| Location | Result within 5000 m of correct |

### 3.2 Training

At the start of the project, we gathered a training set composed of nine different event sites found in the Lund and Malmö area, Sweden. With the help of the training set, we could change the rules or add new ones and easily monitor their overall effect. This concerned both the rules of the annotator, scraper, and the location lookup.

### 3.3 Evaluation

In order to evaluate the system, we gathered a test set of nine, previously unseen, event web sites. The goal was to extract information about all (max. 30) events. The tests were conducted in three parts.

1. In the first part, we used the generic scraper (Sect. 2.2);
2. In the second one, we built simple scrapers (Sect. 2.1) for each of the test sites.
3. We extracted the events manually by hand in the third part.

The results from the first two parts were then compared against the third.

The generic scraper and the simple scrapers were compared in how accurately they extracted the title, date, time, and location of the event. The time of the setup was also compared for both the generic and simple scrapers.

We built a simple scraper for each site specifically to extract the text containing the title, date, time, and the location. The text strings containing the dates and times were then sent to the same algorithm that the generic scraper uses to parse the date and time. Once the text containing the location is extracted, we use the same location lookup in all the scrapers.

### 3.4 Bias Between the Training and Test Sets

The sites in the training set were all composed of a list with events where all the necessary information (title, date, time, location) could be found. In the

**Table 2.** $F_1$ score for full and partial match on test data for the generic scraper.

| | Full | | | | | Partial | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Site | Title | Date | Time | Location | Average | Title | Date | Time | Location | Average |
| lu | 0.0 | 0.967 | 0.767 | 0.433 | 0.542 | 0.4 | 0.967 | 0.933 | 0.633 | 0.733 |
| mah | 0.068 | 1.0 | 0.0 | 0.6 | 0.417 | 0.915 | 1.0 | 1.0 | 1.0 | 0.979 |
| babel | 0.0 | 0.818 | 0.0 | 1.0 | 0.830 | 1.0 | 0.909 | 0.818 | 1.0 | 0.932 |
| lund.cc | 1.0 | 0.667 | 1.0 | 0.652 | 0.714 | 1.0 | 0.967 | 1.0 | 0.652 | 0.905 |
| möllan | 0.0 | 0.857 | 1.0 | 1.0 | 0.75 | 0.0 | 0.857 | 1.0 | 1.0 | 0.714 |
| nsf | 1.0 | 1.0 | 1.0 | 0.0 | 0.673 | 1.0 | 1.0 | 1.0 | 0.286 | 0.822 |
| malmö.com | 1.0 | 1.0 | 0 | 0.691 | 0.543 | 1.0 | 1.0 | 0 | 0.963 | 0.741 |
| burlöv | 0.889 | 0.75 | 0.333 | 0.2 | 0.369 | 1.0 | 0.875 | 0.333 | 0.2 | 0.602 |
| dsek | 0.0 | 0.2 | 0.444 | 0.833 | 0.588 | 1.0 | 0.2 | 1.0 | 0.833 | 0.758 |
| Average $F_1$ | 0.440 | 0.807 | 0.505 | 0.601 | 0.603 | 0.813 | 0.864 | 0.787 | 0.730 | 0.799 |

**Table 3.** $F_1$ score for full match on test data for the generic scraper without loading the event details page.

| | Full | | | | Partial | | | |
|---|---|---|---|---|---|---|---|---|
| Site | Title | Date | Time | Location | Title | Date | Time | Location |
| lu | 1.0 | 1.0 | 0.967 | N/A | 1.0 | 1.0 | 0.967 | N/A |
| mah | 0.967 | 0.929 | 1.0 | N/A | 0.967 | 0.929 | 1.0 | N/A |
| babel | 0.0 | 0.0 | N/A | 1.0 | 1.0 | 0.0 | N/A | 1.0 |

**Table 4.** $F_1$ score for full and partial match on test data for the simple scraper.

| | Full | | | | | Partial | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Site | Title | Date | Time | Location | Average | Title | Date | Time | Location | Average |
| lu | 1.0 | 0.967 | 0.967 | 0.267 | 0.800 | 1.0 | 1.0 | 1.0 | 0.667 | 0.917 |
| mah | 1.0 | 1.0 | 0.0 | 0.7 | 0.675 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| babel | 0.0 | 0.7 | 0.211 | 1.0 | 0.478 | 1.0 | 0.7 | 0.632 | 1.0 | 0.833 |
| lund.cc | 1.0 | 0.667 | 1.0 | 0.622 | 0.822 | 1.0 | 0.967 | 1.0 | 0.622 | 0.897 |
| möllan | 0.857 | 0.667 | 1.0 | 1.0 | 0.881 | 1.0 | 0.833 | 1.0 | 1.0 | 0.959 |
| nsf | 1.0 | 1.0 | 1.0 | 0.0 | 0.75 | 1.0 | 1.0 | 1.0 | 0.0 | 0.75 |
| malmö.com | 1.0 | 1.0 | 0.0 | 0.823 | 0.706 | 1.0 | 1.0 | 0 | 0.912 | 0.728 |
| burlöv | 1.0 | 1.0 | 0.0 | 0.0 | 0.5 | 1.0 | 1.0 | 0.0 | 0.0 | 0.5 |
| dsek | 0.952 | 0.706 | 0.778 | 1.0 | 0.859 | 0.952 | 0.706 | 0.889 | 1.0 | 0.887 |
| Average $F_1$ | 0.868 | 0.856 | 0.551 | 0.601 | 0.719 | 0.995 | 0.912 | 0.725 | 0.689 | 0.83 |

**Table 5.** Time taken for the setup for the test sites.

| Site | Generic | Simple | Manual |
|---|---|---|---|
| lu | 23 min | 83 min | 60 min |
| mah | 7 min | 24 min | 68 min |
| babel | 11 min | 59 min | 15 min |
| lund.cc | 9 min | 13 min | 60 min |
| möllan | 2 min | 31 min | 13 min |
| nsf | 5 min | 24 min | 15 min |
| malmö.com | 31 min | 63 min | 35 min |
| burlöv | 10 min | 30 min | 22 min |
| dsek | 11 min | 23 min | 21 min |
| Average | 12 min | 39 min | 34 min |

test set, most of the sites had a structure that did not have all the required information: Each event had a separate page with all the information, the event details page. The information on the event details page was not composed of the typical compact structured form but rather had more body text. Of the nine sites in the test set, three sites (lund.cc, nsf, dsek) did not require an event details page for the necessary information. But the information on the sites nsf and dsek were in their structure more comparable to a body text. A concept to handle this is presented in Sect. 4.1 that concerns the extraction of the title.

## 4 Conclusion

The setup for the generic scraper took on average 12 minutes, compared to creating a simple scraper for each site that took on average 39 minutes (Table 5). The setup for the generic scraper is more than three times faster than creating a simple scraper for each site. This can be compared to the pure manual labor which took on average 34 minutes per site, thus both scrapers essentially have a pay back time of one pass.

### 4.1 Title

The generic scraper performs rather poorly on the test set while it shows better results on the training set. This is either due to a training overfit or a significant mismatch between the training and test sites. Sect. 3.4 analyzes the mistakes and discusses this problem. When using the system on these pages without loading, they do yield better results, as shown in Table 3. The rest of the failing test sites failed because the system looked to much in the structure where it should have analyzed the layout instead, i.e. it chose links when it should have chosen the ones which were more visually prominent.

### 4.2 Date

The simple scraper is 5% better on the date identification than the generic scraper on average for both the full and partial matches. Examining the scores

for the full match more closely, (Tables 2 and 4), the score for the generic is the same or better than the score for the simple scraper for every site except burlöv and dsek. We even observe a complete failure for dsek. We investigated it and we discovered that dsek expressed the dates relative to the current date e.g. *today*, *tomorrow*. This wasn't implemented yet which made the generic scraper pick another strategy for picking dates, as a result the correct dates were forfeited.

### 4.3 Time

The average scores for the time extraction between the generic and the simple scrapers are rather similar. The system does finds the correct times but does report many false positives, which according to the scoring set in Sect. 3.1 yields only a partial match. The system tends to over detect times. We programmed it to prefer times coupled with dates over solitary times but in the test set, it seems it was rather common to have time and dates further apart. This makes the system choose all times, where it should have chosen a subset. Another pattern was also found: for some sites, the system returned both start and end time separately which shows that the system is lacking rules to bind start and end times together.

### 4.4 Location

The difference between simple and generic scraper is negligible and the problem of location is less about selection and more about actually find and understand the named locations (Tables 2 and 4). The system uses assumed knowledge to fill in what is left out of the events, i.e. knows city, region or location which it can use to fallback to or base the search around. Using this assumed knowledge has proved useful when looking at babel, möllan, dsek, lu and mah and this should hold true on all hyperlocal websites. Even if the system has some basic knowledge about the web page, the location annotation and selection still has problems with disambiguation. This disambiguation problem is partly rooted in the fact that the named locations are within the domain knowledge of the site. As an example, a university website might write lecture halls or class rooms as the location of the event. These named locations could have the same name as pub in another city, a scientist or simply nonexistent in any named location database.

### 4.5 Final Words

At the end of the test cycle, however, we considered that an generic scraper is not only possible to do, but in some cases even better than a simple one. The hardest problem with scraping sites is not necessarily to understand the structure, even if vague. The problem for a scraper is rather to understand what can only be described as domain knowledge. Sites uses a lot of assumed knowledge which can be hard to understand for a machine or even if its understanding could be

completely wrong in the context. For example, lecture halls can be named the same as a pub in the same region, making it hard for a system to determine if the location is correct or not. This might be attainable with better heuristics, e.g. if the location lookup can be made with some hierarchical solution and domain knowledge can be extracted from the sites prior to the extraction of events.

## 5 Future Work

### 5.1 Page Classification

On the Internet, sites show a significant variation and most of them do not contain entertainment events. Therefore a first step in a generic system, the dashed box "Classify" in Figure 1, would be to identify *if* the input web page contains events. If it does not, it makes no sense to scrape it and doing so could even lead to false positives. If web pages could be classified with reasonable certainty, it could also be used with a crawler to create an endless supply of event pages to scrape.

### 5.2 Exploring Repetitiveness

To solve the dashed box "Identify the event list" shown in Figure 1, we investigated the repetitiveness of the event list. With the help of weighing in structural elements, e.g. P, STRONG, H3, it yielded some interesting results on small sites. This technique can potentially be further refined by calibrating weights if the page is annotated using what is described in Sect. 2.3.

### 5.3 Rank and Select with Help of Layout

While the system uses a very limited rank and selection based on an implied layout for title (prefer H3, H2 etc. over raw text), it would be interesting to have the selection fully use layouts. To attract attention and to create desire, the vital information about an event are among the first things the reader is supposed to notice and comprehend. Thus it is usually presented in a visually distinguishing way. This can be achieved by coloring the text differently, making it larger, or simply in a different font or typing. This layout is bundled within the HTML document, possibly modified by the CSS, thus looking at these clues with some heuristics allows to find the visually distinguishing sentences [7]. As an example, an event might use a H3 element for the title, bold for the location, or it might have another background color for the date. If the entire system would use layout to aid the selection we believe that the system will perform better and will yield less false positives.

## References

1. Hogenboom, F., Frasincar, F., Kaymak, U., de Jong, F.: An Overview of Event Extraction from Text. In van Erp, M., van Hage, W.R., Hollink, L., Jameson, A., Troncy, R., eds.: Workshop on Detection, Representation, and Exploitation of Events in the Semantic Web (DeRiVE 2011) at Tenth International Semantic Web Conference (ISWC 2011). Volume 779 of CEUR Workshop Proceedings., CEUR-WS.org (2011) 48–57
2. Liu, M., Liu, Y., Xiang, L., Chen, X., Yang, Q.: Extracting key entities and significant events from online daily news. In Fyfe, C., Kim, D., Lee, S.Y., Yin, H., eds.: Intelligent Data Engineering and Automated Learning - IDEAL 2008. Volume 5326 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2008) 201–209
3. Chun, H.w., Hwang, Y.s., Rim, H.C.: Unsupervised event extraction from biomedical literature using co-occurrence information and basic patterns. In: Proceedings of the First international joint conference on Natural Language Processing. IJCNLP'04, Berlin, Heidelberg, Springer-Verlag (2005) 777–786
4. Exner, P., Nugues, P.: Using Semantic Role Labeling to Extract Events from Wikipedia. In van Erp, M., van Hage, W.R., Hollink, L., Jameson, A., Troncy, R., eds.: Workshop on Detection, Representation, and Exploitation of Events in the Semantic Web (DeRiVE 2011) at Tenth International Semantic Web Conference (ISWC 2011). Volume 779 of CEUR Workshop Proceedings., CEUR-WS.org (2011) 38–47
5. Borsje, J., Hogenboom, F., Frasincar, F.: Semi-automatic financial events discovery based on lexico-semantic patterns. Int. J. Web Eng. Technol. **6**(2) (January 2010) 115–140
6. Hienert, D., Luciano, F.: Extraction of historical events from wikipedia. In: Proceedings of the First International Workshop on Knowledge Discovery and Data Mining Meets Linked Open Data, CEUR-WS.org (2012)
7. Cai, D., Yu, S., Wen, J.R., Ma, W.Y.: Extracting content structure for web pages based on visual representation. In: Proceedings of the 5th Asia-Pacific web conference on Web technologies and applications. APWeb'03, Berlin, Heidelberg, Springer-Verlag (2003) 406–417