µRaptor: A DOM-based system with appetite for hCard elements

Emir Muñoz[1,2], Luca Costabello[1], and Pierre-Yves Vandenbussche[1]

[1] Fujitsu Ireland Limited
[2] National University of Ireland, Galway
E-mail: Emir.Munoz@ie.fujitsu.com

Abstract. This paper describes µRaptor, a DOM-based method to extract hCard microformats from HTML pages stripped of microformat markup. µRaptor extracts DOM sub-trees, converts them into rules, and uses them to extract hCard microformats. Besides, we use co-occurring CSS classes to improve the overall precision. Results on train data show 0.96 precision and 0.83 F_1 measure by considering only the most common tree patterns. Furthermore, we propose the adoption of additional constraint rules on the values of hCard elements to further improve the extraction.

1 Introduction

In this paper we present our contribution to the Linked Data for Information Extraction Challenge[1] at the LD4IE 2014 Workshop. The challenge aims at using embedded structured data in web pages, e.g., RDFa, Microformats, or Microdata, to bootstrap and train supervised systems to extract structured data from the Web. Current web pages information extraction techniques focus on the (automatic) induction of wrappers to scrape meaningful parts (see [2] for a survey). However, by extracting all the properties contained in hCards[2], such techniques achieve low precision and recall. [1] reports a F-score of 0.59 extracting person attributes from plain web pages. By relying on (X)HTML pages with hCard markup, µRaptor identifies hCard-like sections in (X)HTML documents without the need for hCard markup, and generates a corresponding RDF representation.

Extracting hCard information from (X)HTML pages presents two main challenges: 1) the identification of hCard sections in non semantically annotated (X)HTML pages, and 2) the qualification of hCard elements type (e.g. `family-name`, `given-name`). Our system addresses the two challenges by using the training document set to extract DOM sub-trees containing assessed hCard description and generating CSS selector rules. To increase the CSS selector rules precision, we generate a co-occurrence matrix of CSS classes and hCard properties. This also enhances the qualification of hCard property types. Furthermore, using value constraints rules associated to hCard properties, we assess the probability for an HTML element to be of a particular hCard type.

[1] http://data.dws.informatik.uni-mannheim.de/LD4IE/
[2] http://microformats.org/wiki/hcard. A microformat used for publishing, people, companies, and organizations on the Web, using vCard properties.

2 µRaptor System Description

Our system is composed of two distinct phases: The first phase aims at training the Information Extraction system (i.e. it learns the tree structures of hCard in the train set DOM); the second phase targets hCard semantic markup extraction. A flow diagram of our system is illustrated in Figure 1. Moreover, the source code of µRaptor is publicly available at https://github.com/emir-munoz/uraptor, and works with the data provided in the Challenge website[3].

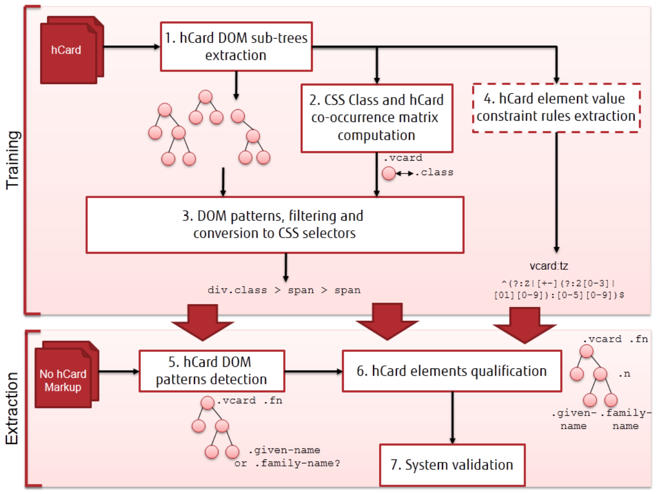
Training
hCard
1. hCard DOM sub-trees extraction
2. CSS Class and hCard co-occurrence matrix computation
4. hCard element value constraint rules extraction
.vcard
.class
3. DOM patterns, filtering and conversion to CSS selectors
div.class > span > span
vcard:tz
^(?:Z|[+-](?:2[0-3]|[01][0-9]):[0-5][0-9])$
Extraction
No hCard Markup
5. hCard DOM patterns detection
6. hCard elements qualification
.vcard .fn
.given-name or .family-name?
7. System validation
.vcard .fn
.n
.given-name
.family-name

Fig. 1: µRaptor system architecture with training and evaluation parts.

2.1 Training Phase

1) hCard DOM sub-trees extraction. The training phase of our system takes as input the training set composed of 9,386 HTML documents containing hCard semantic markup. First, we clean the input HTML pages by removing all non-content related tags (e.g., style, JavaScript). Then, for each document, hCard sections are extracted using the `[class*=vcard]` CSS selector, returning sub-trees of the original DOM tree containing the hCard markup.

2) CSS classes and hCard co-occurrence matrix computation. HTML elements annotated with hCard markup may also be annotated with other CSS classes. To achieve higher precision during the extraction phase, we compute a matrix of hCard element and other CSS class co-occurrences that will help refining the extraction rules and qualifying the hCard elements type.

3) DOM patterns, filtering, and conversion to CSS selectors. Once all hCard DOM sub-trees have been extracted, we count their frequencies. We keep only the most frequent patterns (thus achieving higher extraction precision) and

[3] http://data.dws.informatik.uni-mannheim.de/LD4IE/data/

convert them into rules to guide the extraction of hCards in other documents during the extraction phase. This conversion is done by transforming the extracted DOM sub-trees into CSS selectors[4] using the *child combinator* operator[5]. Since tree-like structures cannot be directly expressed with CSS selectors, we prune each DOM sub-tree, leaving only the longest path, and we convert such path to a chain of CSS child combinators (see example in Figure 2). Co-occurring classes are used at this stage to increase the precision of the extraction rules. For instance, in the rule with selector "`div > img + cite`" we found that the most common co-occurring class with `vcard` annotation is `comment-author`. Hence, we modify the rule to "`div.comment-author > img + cite`".

4) hCard element value constraint rules extraction. Sometimes the extracted hCard microformats contain erroneous data (e.g. phone numbers in `class="email"` annotations). Although not implemented in the current µRaptor version, we propose to detect and fix such inconsistencies with a technique that we introduced in [3]. Such strategy creates content patterns that can be used as constrains to validate hCard properties extraction.

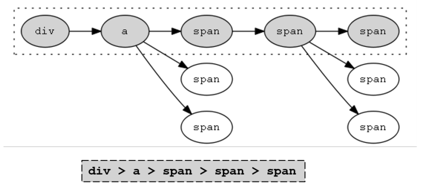

Fig. 2: A sample DOM-subtree to CSS selector conversion.

2.2 Extraction Phase

5) hCard DOM patterns detection. During the extraction phase, we run each of the extracted rules over the test set of documents (containing no hCard markup) to identify hCard candidates. Rules are applied in a specific order: From the most long/specific to the most short/general, according to their CSS selectors. We add the constraint that only one rule will visit a hCard candidate.

6) hCard elements qualification. Once a DOM sub-tree containing a hCard candidate is extracted, we modify its DOM elements structure to add the corresponding hCard markup as captured during the training phase. In other words, we assign hCard properties such as `vcard`, `fn`, `adr`, `email`, and so forth, to elements in the DOM. Finally, we generate the N-Quads output file using Apache Any23[6] with the HTML document URI as graph name (context).

3 Results and Discussions

µRaptor evaluation (Figure 1, step 7) was performed against the gold standard provided built using Apache Any23. We use Information Retrieval measures,

[4] http://www.w3.org/TR/css3-selectors/
[5] http://www.w3.org/TR/css3-selectors/#child-combinators
[6] http://any23.org

namely Precision (P), Recall (R) and F_1-score to evaluate the performance of our system. Let be $A = \{\text{gold standard n-quads}\}$, and $B = \{\text{µRaptor n-quads}\}$, then

$$P = \frac{|A| \cap |B|}{|B|},\ R = \frac{|A| \cap |B|}{|A|},\ \text{and}\ F_1 = \frac{2PR}{P+R}$$

We report our results over training and testing data in Table 1. A higher recall can be reached by adding more rules at the cost of overfitting the system.

Table 1: µRaptor results achieved in datasets.

Dataset	P	R	F_1
Train	0.96	0.73	0.83
Test	0.92	0.67	0.77
Average	0.94	0.7	0.8

During our analysis, the first interesting observation is that the occurrences of hCard properties in the training dataset follow a logarithmic distribution (see Figure 3). Aside from `rdf:type` (that occurs in 35.2% of cases), the most popular properties are `vcard:n` (16.5%), `vcard:fn` (10.2%), `vcard:given-name` (10.1%), `vcard:family-name` (9.9%). The remaining properties account for a 18.1% and consists in the long tail of the distribution (e.g. `vcard:role` occurs 32 times only, `vcard:honorific-prefix` only once). To extract the latest properties, we could define specific rules to extract the less frequent cases. However, we did not consider less frequent cases, thus affecting our overall recall. This has been observed in the results given by the Challenge organizers after evaluation over the test set. Less frequent properties are not found by the current set of rules, adversely affecting the overall recall. Again, µRaptor can easily be extended with more rules to cover those cases improving the performance of the system.

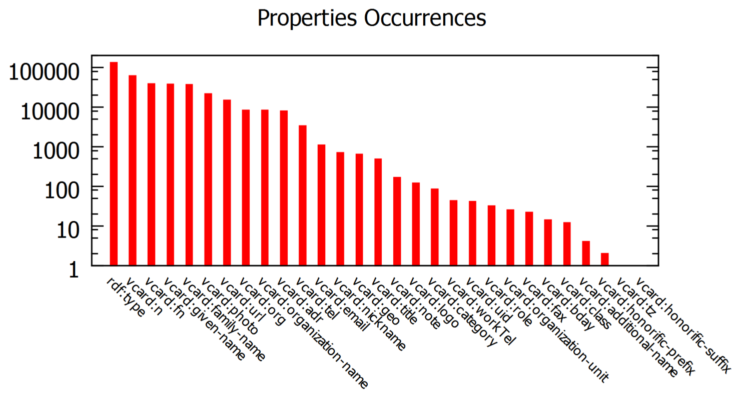

Fig. 3: Properties occurrences in training data (note the logarithmmic scale).

Surprisingly, we notice that the gold standard provided for the challenge does not include a number of `hCard` microformats, due to an Apache Any23 inability to detect a certain number of them while parsing (X)HTML pages (however, the extraction is successful if only the HTML snippet is provided). For instance, the following microformat[7] in `train1` dataset has not been extracted:

```
<span class="post-author vcard"> Posted by
  <span class="fn" itemprop="author" itemscope="itemscope"
        itemtype="http://schema.org/Person">
    <meta content="https://plus.google.com/117423557913834377613"
        itemprop="url">
    <a href="https://plus.google.com/117423557913834377613"
        rel="author" title="author profile">
    <span itemprop="name">Bianca Swanepoel</span>
    </a>
  </span>
</span>
```

µRaptor, on the other hand, extracts more RDF n-quads than the gold standard, thus affecting the precision (we attribute this to the preliminary (X)HTML pages cleaning, Figure 1, step 1). The cleaning phase removes non relevant and noisy parts in the (X)HTML allowing a better extraction. A manual inspection found that our “extra” N-Quads are valid extractions.

4 Conclusion

µRaptor is a rule-based system that extracts structured information such as hCards, relying on (X)HTML pages DOM model. By considering only the 30 most frequent tree patterns for hCards (out of ca. 700), we achieved 0.94 of average precision. Experiments determined that adding few more rules produces a marginal difference in F_1-measure. Despite our approach performs well on training and testing data, µRaptor does not cover all hCard properties, since not all of them are used by web designers or content management systems (e.g., Drupal), making challenging to extract patterns to identify them.

Acknowledgments. This work has been supported by KI2NA project funded by Fujitsu Laboratories Limited and Insight Centre for Data Analytics at NUI Galway (formerly known as DERI Galway).

References

1. T. István Nagy. Person attribute extraction from the textual parts of web pages. *Acta Cybern.*, 20(3):419–439, Aug. 2012.
2. B. Liu. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. Data-Centric Systems and Applications. Springer, 2011.
3. E. Muñoz. Learning Content Patterns from Linked Data. In *Linked Data for Information Extraction (LD4IE) Workshop, ISWC*. CEUR, 2014.

[7] Snippet included in `http://www.themigratingswans.blogspot.ie/2013/12/december-3.html`