Information Extraction from Web Sources based on Multi-aspect Content Analysis

Martin Milicka and Radek Burget

Faculty of Information Technology, IT4Innovations Centre of Excellence
Brno University of Technology, Bozetechova 2, 612 66 Brno, Czech Republic
{imilicka,burgetr}@fit.vutbr.cz

Abstract. Information extraction from web pages is often recognized as a difficult task mainly due to the loose structure and insufficient semantic annotation of their HTML code. Since the web pages are primarily created for being viewed by human readers, their authors usually do not pay much attention to the structure and even validity of the HTML code itself. The CEUR Workshop Proceedings pages are a good illustration of this. Their code varies from an invalid HTML markup to fully valid and semantically annotated documents while preserving a kind of unified visual presentation of the contents. In this paper, as a contribution to the ESWC 2015 Semantic Publishing Challenge, we present an information extraction approach based on analyzing the rendered pages rather than their code. The documents are represented by an RDF-based model that allows to combine the results of different page analysis methods such as layout analysis and the visual and textual feature classification. This allows to specify a set of generic rules for extracting a particular information from the page independently on its code.

Keywords: document modeling, information extraction, page segmentation, content classification, ontology, RDF

1 Introduction

The documents available on the web present a large and ever growing source of information. However, extracting information from the HTML documents remains a challenging tasks mainly because of the high variability of the markup, loose structure of the documents and very rare use of any kind of semantic annotations that could be used for recognizing a particular information in the document.

The research in this area includes many different approaches including a direct HTML code analysis by different methods [7, 8], DOM analysis [6], page layout [2] or other visual feature analysis [10]. As the research results show, the web documents are too variable for the a simple and straightforward solution. The document processing cannot be based only on single aspect such as the text content, visual features or document structure because each approach is suitable for a different kind of documents. Therefore, we propose an approach that can combine multiple aspects of the document.

The documents may be described on different levels of abstraction starting with the code through the rendered page layout and visual features of the contents to a logical structure as it is expected to be interpreted by a human reader. We propose an ontology-based document model that is able to capture all the mentioned kinds of information. For each level of the description, we use a specific ontology. The highest abstraction level represents the target domain of the extracted information.

In this paper, we apply this approach to the processing of the CEUR Workshop proceedings as a part of the ESWC 2015 Semantic Publishing Challenge. We employ a combination of algorithms such as page segmentation or content classification for building the proposed model from source documents. Based on a combination of different features, we propose the way of extracting the logical structure of the document. This structure is finally transformed to the specific domain ontology. This approach allows to abstract from the HTML implementation details and increase the robustness of the extraction.

2 System Architecture

The presented information extraction system is based on our recently developed FITLayout[1] framework [9] – a generic framework for web page segmentation and its further analysis. The complete architecture overview is shown in Fig. 1. Implementation details specific for the Semantic Publishing Challenge 2015 are described later in section 4.

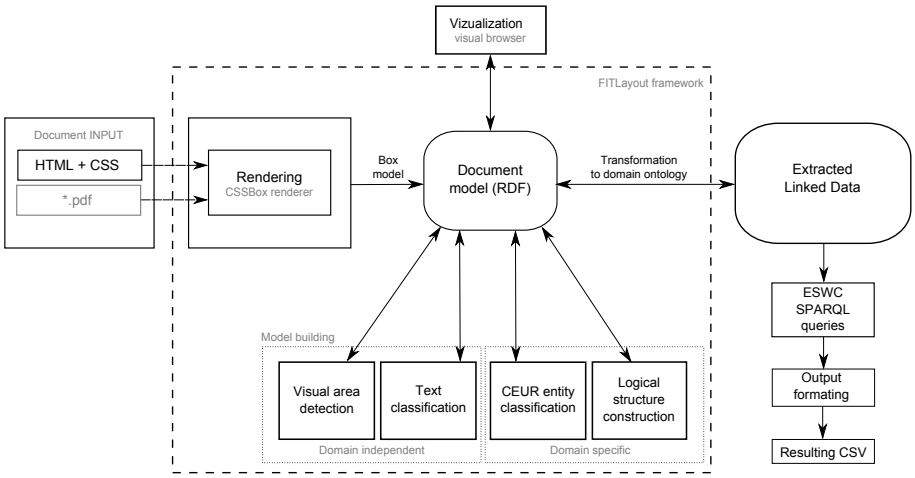

Fig. 1. Extraction System Architecture

[1] http://www.fit.vutbr.cz/~burgetr/FITLayout/

Unlike most existing information extraction systems, our system does not analyze the HTML or CSS code of the input documents directly. Instead, it operates on the rendered page trying to use the same information as the user who is actually browsing the page. This allows to abstract from the HTML-related problems such as irregular code structure, invalid markup, etc.

The individual documents (CEUR pages) are processed independently on each other. The processing consists of several steps. The results of each step are stored to an internal RDF repository; each step adds more information to the model of the processed document. First, source pages are rendered using a built-in rendering engine that provides the information about the layout and visual features of the individual pieces of the contents. Additionally, basic text analysis steps are applied on the document in order to recognize important entities in the text such as dates, times, capitalized sequences or personal names. Subsequently, the obtained model is analyzed and the desired information such as editors, paper titles, authors, etc. is recognized using a set of quite simple rules based on the actual presentation of the individual content parts. Based on the recognized parts of the contained information, we build a *logical structure* of the document that represents the semantic relationships. Finally, this structure is transformed to the resulting linked data set.

2.1 Page Rendering

The rendering engine processes the input HTML and the linked CSS files and produces the information about the content layout in the page. The layout is represented by a *box model* generally defined in the CSS specification [1]. This model describes the positions of the individual pieces of content in the resulting page and their further visual properties (fonts, colors, etc.) Each box corresponds to a rectangular area in the rendered page. The boxes are organized in a hierarchical structure called a *box tree* that roughly corresponds to the source DOM structure.

The obtained box tree is transformed to RDF data using the FITLayout box model ontology described in section 3.1. In the subsequent steps of the model building, more information is added to the page model as the result of the individual analysis methods.

2.2 Model Building

The model building phase consists of four analysis steps. The first two of them are domain-independent; they are not specific for the SemPub2015 task. The other two steps are specific for the target domain. The details of the individual steps are described later in section 4.

1. **Visual area detection.** We identify all the boxes in the box tree that are visually distinguishable in the resulting pages. These boxes form the basic *visual areas*. We construct a tree of visual areas based on their visual nesting

in the rendered page. The resulting area tree is described using the corresponding FITLayout segmentation ontology (see section 3.2). Later, each area may be assigned any number of text *tags* that represent the expected meaning of the area at different levels.

2. **Text classification.** We go through the leaf areas of the visual area tree and we identify important generic entities in the text such as dates, times or personal numbers. Based on the discovered entities, we assign tags to the areas that indicate the type of their content.
3. **CEUR entity classification.** Based on the previous two steps, i.e. the layout model and the properties of the text, we identify the CEUR entities such as the workshop title, editor names, paper titles and authors, etc. Their discovery is based on mutual positions of the corresponding areas and regular patterns in the presentation styles. The areas that correspond to the individual CEUR entities are again marked by the appropriate tags. For example, a visual area that obtained a *persons* tag in the previous text classification step (i.e. it contains some personal names) is likely to obtain the *editors* or *authors* tag depending on where the area is placed within the page.
4. **Logical structure construction.** The purpose of the logical structure is to represent the relationships among the CEUR entities tagged in the previous steps. For example, the title, authors and page numbers that belong to a single paper, papers that belong to a single section, etc. In a domain-dependent way, we transform the tagged area tree to the logical structure tree where the logical nodes correspond to particular text strings (e.g. the names themselves) and the parent-child relationships correspond to the semantic subordination of the entities (e.g. the title, authors and pages are child nodes of a paper node). Each node is marked with a single tag that specifies its semantic.

The whole process corresponds to the transition from the rendered page model (the box tree) through the page layout model (the visual area tree) to its semantic interpretation (the logical area tree). In the next step, the resulting logical model can be transformed to the target domain ontology.

2.3 Output Dataset Generation

The resulting logical structure tree that is obtained from the model building phase and stored in the intrenal RDF repository contains the complete information extracted from the source page together with its structure. The output dataset generation only consists of transforming the data represented using the FITLayout internal visual area ontology to the target domain ontology described in section 3.5. This is implemented as a single SPARQL query[2] on the internal RDF repository.

[2] `https://github.com/FitLayout/ToolsEswc/blob/master/sparql/logicalTree2domain.sparql`

3 Ontological Model

The ontological model describes the processed document at multiple levels of abstraction. We have defined five abstraction levels of document description where each higher level adds specific knowledge to the previous one. Each level of description is characterized by its ontology. The hierarchy of levels is shown in Fig. 2. We can see that all the levels can be divided in two groups: domain-independent and domain-specific. The tagging level in the middle joins the two parts together.

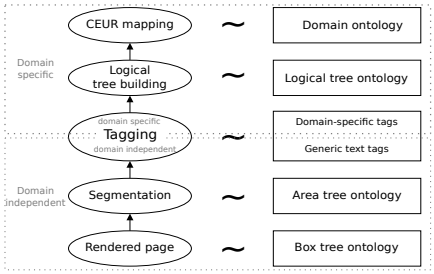

Fig. 2. Ontological model

3.1 Rendered Page Level

At the level of the rendered page, the ontology-based model corresponds to the document box model where its rendering is based on the source data presented in the HTML document and visual features defined by Cascading Style Sheets (CSS).

The schema of the presented *Box model ontology* is on Fig. 3 A). Every class is based on the *Rectangle* class which defines characteristic size, position and visual features. A *Box* denotes a base displayed document element. It follows the definition from the CSS formatting model [1]. The boxes may be nested, which creates a hierarchical structure similar to the Document Object Model (DOM). The *Page* class represents the whole rendered page. The *belongsTo* property denotes the relationship between the Page and some rectangular objects (boxes) that create the contents of the page. The *Box* can be further specialized into the *ContainerBox* or *ContentBox* classes where the *ContainerBox* may contain other boxes (allows nesting). The *ContentBox* represents a Box that contains a connections of content objects like images, textual information or common objects like Flash, video, etc.

[3] `b: http://fitlayout.github.io/ontology/render.owl#`
[4] `a: http://fitlayout.github.io/ontology/segmentation.owl#`

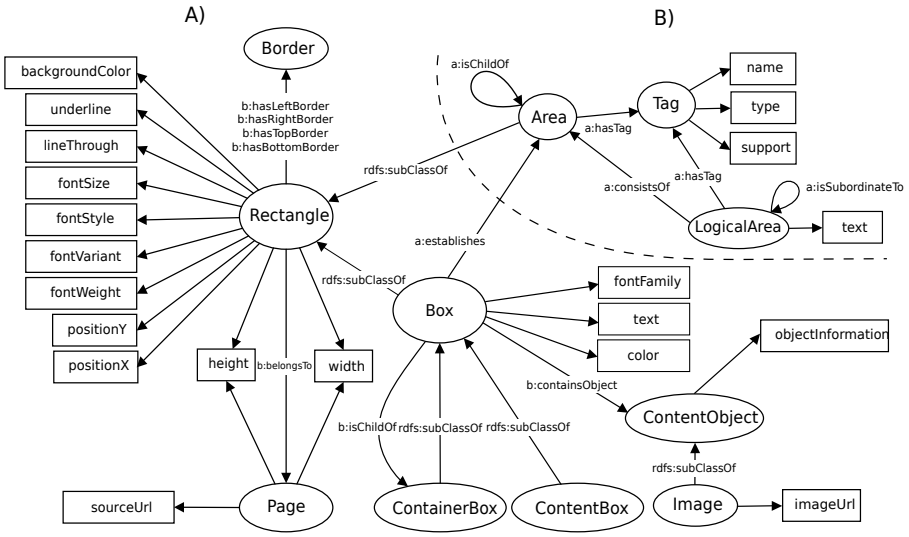

Fig. 3. A) Box tree ontology[3] B) Area based ontology[4]

3.2 Segmentation Level

Page segmentation generally detects the visually distinguished segments in the rendered page (we call them *visual areas* in this paper). There exist many page segmentation algorithms; one of the most popular ones is called VIPS [4].

The segmentation model extends the Box model by a possibility of representing the visual areas. In the figure 3 B) we can see a part of segmentation ontology design. The basic *Area* class is defined as a specialization of the *Rectangle* class from the Box model ontology. It represents the visual areas detected during the page segmentation. A visual area is usually created by a collection of boxes contained in this visual segment. Visual areas may be nested and create a hierarchy based on their visual nesting similarly to boxes.

3.3 Tagging Level

The tags are represented by the *Tag* class (in Fig. 3 B)); multiple tags may be assigned to a single visual area. Each tag is represented by its *name* and *type* where the type represents a set of tags with the same purpose (e.g. the tags obtained from text classification) and the name corresponds to the actual tag value.

In section 4, we give an overview of the tags used for the given domain. Some of them are domain-independent (Table 1), some are domain-dependent (Table 2).

3.4 Logical Tree Level

The logical structure represents the actual interpretation of the tagged visual areas in the target domain. Each logical area corresponds to a semantic entity identified as a text string contained in some visual areas (e.g. an author name). It is represented by the *LogicalArea* class in (Fig. 3). Each logical area has a single tag assigned that denotes its meaning in the target domain (e.g. a paper title).

The logical areas are organized to a hierarchical structure again (using the *isSubordinateTo* property). However, unlike the visual areas, where the hierarchy represents the visual nesting, for logical areas, the hierarchy corresponds to the logical relationships among the entities – e.g. a paper and its title or page numbers.

The resulting logical area tree provides a generic representation of the extracted information and its structure and it can be directly mapped to the target domain ontology.

3.5 Domain Level

The domain ontology defines the entities and their properties in the target domain. It is used for the resulting data set produced by our extraction tool. For the the CEUR proceedings domain, we use a combination of existing ontologies shown in Fig. 4 that is greatly inspired by [8] with some simplifications.

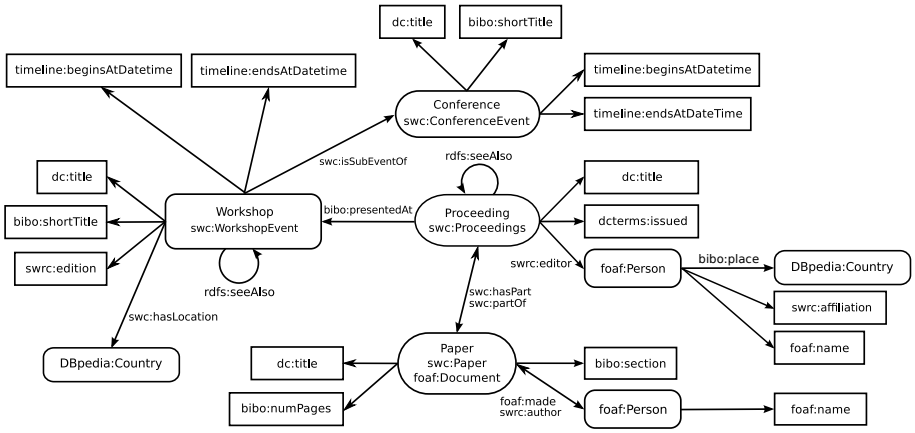

Fig. 4. Domain ontology - ESWC proceedings

4 System Implementation

The FITLayout framework used as a base for our system implements a variety of general tasks such as page rendering, page segmentation and text feature anal-

ysis. Moreover, it allows to implement custom extensions and add them to the page processing chain. For the purpose of the CEUR proceedings processing, we have implemented several domain-specific extensions that include the CEUR entity recognition and a custom logical structure builder specific for this particular task[5].

We made several experiments with using the microformats available in some of the CEUR volume pages for training a visual feature classifier that would be later used for the remaining volumes. However, the presentation style of the individual volumes is quite variable in terms of the used fonts, layout or information ordering. Therefore, we have decided to process the individual pages independently. In the final version of our tools, we do not use any kind of classifier training (apart from the pre-trained Stanford NER classifier used for recognizing the personal names as described in section 4.2). Instead of this, we just gather statistic about the frequently used presentation patterns and styles used in the currently processed page and we assume the most frequent one to be consistently used in the page as described in section 4.3. The microformats are not used at all in the end because their availability is not guaranteed.

In the following sections, we explain the most important details of the whole information extraction process.

4.1 Layout Analysis

The FITLayout framework assumes a usage of a page segmentation method for the construction of the visual area tree. However, due to the relatively simple layout of the CEUR proceedings, we decided not to use a full-featured page segmentation algorithm. Instead, we just use a basic visual area recognition algorithm that corresponds to the initial step of our previously published page segmentation algorithm [3]. From the box tree obtained from rendering, we choose the boxes that are visually distinguishable in the page: they directly represent a piece of text or image content or they have some visible visual separator: a separating background color or a border around the box.

For the CEUR proceedings, the resulting layout model is usually very flat: Most of the content areas are directly the child nodes of the root node because there is usually no visual nesting used in the layout. The only exception is the title of some of the proceedings that is visually framed.

4.2 Generic Text Tagging

Area tagging is used to roughly classify the visual areas that contain certain kind of information. The FITLayout framework provides a set of general purpose taggers that assign tags of the `FitLayout.TextTag` type to the visual areas by a simple analysis of the contained text mainly using regular expressions. Table 1 describes the text tags we have used for the given task and the way of their assignment to the visual areas.

[5] `https://github.com/FitLayout/ToolsEswc`

Table 1. Tags added during the text feature analysis (tag type `FitLayout.TextTag`)

Tag	Meaning	Way of recognition
dates	A date in recognizable format	Regular expressions and specific keywords (months)
pages	Page span specification	Regular expression
persons	Personal names	Stanford NER classifier [5]
title	Paper title	Regular expression

The used regular expressions are quite general (especially for the paper titles), and the used generic NER classifier is not 100% accurate neither. Therefore, the tag assignment obtained in this phase provides just a rough and approximate classification of the areas. Further refining is performed in the following CEUR entity recognition phase.

4.3 CEUR Entity Recognition

The CEUR entity recognition consists of assigning another set of tags to the discovered visual areas. These tags correspond to the individual types of information that we want to extract from the source document. The complete list of the assigned tags (with the type `ESWC`) and their meaning is in Table 2.

Table 2. Tags used for the CEUR entity annotation (tag type `ESWC`)

Tag	Meaning
vtitle	Volume title
vcountry	Workshop location (country)
veditor	Editor name
vdate	Date(s) of the workshop

Tag	Meaning
subtitle	Volume subtitle (proceedings)
title	Paper title
authors	Paper author(s)
pages	Paper pages

The transition from the general *text tags* listed in Table 1 to the *semantic tags* listed in Table 2 corresponds to the disambiguation and refining of the rough text classification. We assume that some text tags may be missing or may have been assigned incorrectly. Some tags are ambiguous, e.g. the *persons* tag may indicate author or editor names depending on context.

For assigning the semantic tags, our refining algorithms take into account the following aspects:

- *Common visual presentation rules* – there exist some commonly used rules for visual formatting of the presented information in a document. E.g. a title or subtitle is written in larger font or at least bolder than a normal text.
- *Regularity in presentation style* – we assume that all the information of the same meaning (e.g. all paper titles) is presented with the same visual style (fonts, colors, etc.) in a single proceedings page.

- *Regularity in layout* – some proceedings put author names before the paper title, some put them below or on the same line. However, this layout is again consistent through the whole proceedings page.
- *Locality of the information* – information of the same kind is presented in one area of the page. We can identify an area containing editors, papers, etc. The order of these area remains the same in all the proceedings pages.
- *Textual hints* – some key phrases such as "Edited by" or "Table of Contents" are commonly used in most of the proceedings. When they are found in the page, they can be used to refine the expected area where a particular information is located within the page.

Our algorithm works in the following steps:

1. We discover the position of the workshop title and the repeating layout and style patterns that (together with the assigned text tags from Table 1) correspond to the published papers and their authors and similarly for editors and their affiliations.
2. Based on the discovered patterns, we guess approximate areas in the rendered page that are likely to contain a particular information: the workshop title, subtitle (proceedings information), editors, papers and submission details. If the text hints such as "Edited by" are present in the pages, the expected area bounds are adjusted appropriately.
3. In these areas, we find the most frequent font style used for each type of information (e.g. author names) and the most frequent layout pattern (authors before or after the title, etc.) Then, we assign the appropriate semantic tags from Table 2 to all the visual areas using the same font style that correspond to the discovered layout pattern. This solves the possible inaccuracy of the text tag assignment.

The workshop title is discovered by its font size (it's always written with the largest font size used in the page). The editor area is guessed by searching personal names between the workshop title and the papers (the "Table of contents" text may be used as a hint when present) and the subtitle is located between the title and the editors.

As the result, we obtain a refined tagging of the visual areas that indicates their semantics.

4.4 Logical Structure Construction

The last logical structure construction phase has two main goals:

- Extract the text data from the tagged visual areas. The area may contain multiple kinds of information (e.g. several author names, several editors or some additional text that should be omitted).
- Put together the information that belongs to a single entity: the name and affiliation of a single editor or the title, authors and pages of a single paper.

These goals correspond to the construction of a tree of *logical areas* as defined in section 3.4. The text extraction corresponds to the identification of the logical areas and the relationships among the areas (denoted using the *a:isChildOf* property) are used for creating a tree of logical areas where the child nodes specify the properties of its parent node.

We have implemented a custom logical tree builder that goes through the visual area tree and creates the logical areas organized in subtrees depending on the assigned semantic tags. For this, some more text processing is usually required: splitting the author area to several author names by separators, completing the editor affiliations by matching the different kinds of symbols and bullets and extracting the data such as workshop date from longer text lines.

The countries in the editor affiliations are recognized by a simple matching with a fixed list of countries and their DBPedia resource IRIs (a CSV extracted from DBPedia).

The workshop and conference acronym extraction is based on a simple text parser that recognizes all the acronyms and the ordinals in the text. In order to distinguish between the workshop and the conference acronyms, we try to locate the particular keywords (e.g. "colocated with") in the subtitle and we also compare the sets of acronyms found in the title and the subtitle since the conference acronym is very rarely present in the main title.

Some information such as the paper IRIs must be obtained from the underlying code from the `id` or `href` attributes. Therefore, in our stored rendered page model, we maintain the information about the source DOM nodes that produce the given box displayed in the page.

The resulting logical structure description is added to the FITLayout internal RDF repository and it can be directly transformed to the output linked data set by mapping to the target ontology.

4.5 CEUR Index Page Processing

The CEUR proceedings index page is a specific source of information. We use this page for locating the related workshops (the *see also*) information, the date of publication. We also use the volume title from the index page in the final output because the title in the individual pages is slightly different in some cases.

Since the index page is just a single HTML document with a specific style and quite a regular structure, we have just used a simple "old school" Unix `awk` script for extracting this data directly from the HTML code. This script produces a CSV output that is used by the logical tree builder to complete the logical structure.

5 Conclusions

In this paper, we have presented a web information extraction approach based on a complex modelling of different aspects of the processed document. Our system analyzes the rendered document and in multiple steps, it guesses and

later disambiguates the semantics of the individual text parts by combining the page segmentation and text classification methods with specific extraction rules based on visual presentation of the content. This approach allows to avoid HTML-related implementation details. The extraction task is specified on quite a high level of abstraction that ensures the tolerance of the method to different variations in the processed documents.

Acknowledgments

This work was supported by the BUT FIT grant FIT-S-14-2299 and the IT4Innovations Centre of Excellence CZ.1.05/1.1.00/02.0070.

References

1. Bos, B., Lie, H.W., Lilley, C., Jacobs, I.: Cascading Style Sheets, level 2, CSS2 Specification. The World Wide Web Consortium (1998)
2. Burget, R.: Layout based information extraction from HTML documents. In: ICDAR 2007. pp. 624–629. IEEE Computer Society (2007)
3. Burget, R., Rudolfová, I.: Web page element classification based on visual features. In: 1st Asian Conference on Intelligent Information and Database Systems ACIIDS 2009. pp. 67–72. IEEE Computer Society (2009)
4. Cai, D., Yu, S., Wen, J.R., Ma, W.Y.: VIPS: a Vision-based Page Segmentation Algorithm. Microsoft Research (2003)
5. Finkel, J.R., Grenager, T., Manning, C.: Incorporating non-local information into information extraction systems by gibbs sampling. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics. pp. 363–370. ACL '05 (2005)
6. Hong, J.L., Siew, E.G., Egerton, S.: Information extraction for search engines using fast heuristic techniques. Data Knowl. Eng. 69(2), 169–196 (Feb 2010), http://dx.doi.org/10.1016/j.datak.2009.10.002
7. Hong, T.W., Clark, K.L.: Using grammatical inference to automate information extraction from the Web. Lecture Notes in Computer Science 2168, 216+ (2001)
8. Kolchin, M., Kozlov, F.: A template-based information extraction from web sites with unstable markup. In: Presutti, V., Stankovic, M., Cambria, E., Cantador, I., Di Iorio, A., Di Noia, T., Lange, C., Reforgiato Recupero, D., Tordai, A. (eds.) Semantic Web Evaluation Challenge, Communications in Computer and Information Science, vol. 475, pp. 89–94. Springer International Publishing (2014), http://dx.doi.org/10.1007/978-3-319-12024-9_11
9. Milicka, M., Burget, R.: Multi-aspect document content analysis using ontological modelling. In: Proceedings of 9th Workshop on Intelligent and Knowledge Oriented Technologies (WIKT 2014). pp. 9–12. Vydavateľstvo STU (2014)
10. You, Y., Xu, G., Cao, J., Zhang, Y., Huang, G.: Leveraging visual features and hierarchical dependencies for conference information extraction. In: Ishikawa, Y., Li, J., Wang, W., Zhang, R., Zhang, W. (eds.) Web Technologies and Applications, Lecture Notes in Computer Science, vol. 7808, pp. 404–416. Springer Berlin Heidelberg (2013), http://dx.doi.org/10.1007/978-3-642-37401-2_41