Information Extraction from the Web by Matching Visual Presentation Patterns

Radek Burget

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

Abstract. The documents available in the World Wide Web contain large amounts of information presented in tables, lists or other visually regular structures. The published information is however usually not annotated explicitly or implicitly and its interpretation is left on a human reader. This makes the information extraction from web documents a challenging problem. Most existing approaches are based on a top-down approach that proceeds from the larger page regions to individual data records, which depends on different heuristics. We present an opposite bottom-up approach. We roughly identify the smallest data fields in the document and later, we refine this approximation by matching the discovered visual presentation patterns with the expected semantic structure of the extracted information. This approach allows to efficiently extract structured data from heterogeneous documents without any kind of additional annotations as we demonstrate experimentally on various application domains.

Keywords: web data integration, information extraction, structured record extraction, page segmentation, content classification, ontology mapping

1 Introduction

The World Wide Web contains a vast amount of documents containing data records presented in a regular, visually consistent way using different kinds of lists, tables or other logical structures. Typical examples include product data, events, exchange rates, sports results, timetables and many more. Although the structure of the presented information is generally predictable for every application domain, the actual data records may be presented in the HTML documents in countless ways.

For large and consistent data sources such as Wikipedia, it is possible to define extraction templates that may be reused for a great number of pages. However, for heterogeneous sources where every document may use different presentation patterns, this approach is not feasible. The great variability in presentation and almost no semantic annotations available in HTML documents make the automatic integration of such web sources to structured datasets (such as DBPedia) a challenging problem.

In this paper, we present a method for the discovery and extraction of structured records in web documents. In contrast to most current approaches that perform a complex analysis of the document HTML code or its visual organization in order to detect repeating structures (top-down approach) [1, 8, 13, 14], we use an opposite (bottom-up) approach: We start with the smallest consistent text elements and we match the visual relationships among these elements with the expected structure of the extracted records. This way, we are able to automatically discover the visual patterns used for presenting the data records in the given document.

The most important benefits of the presented approach are the following:

- The extraction task specification is based only on a generic domain knowledge consisting of the logical relationships among the individual data fields to be extracted and a very general specification of allowed values for each data field.
- No templates need to be used and no labels or annotations are required in the source documents.
- The method can be easily adapted for any target domain as it allows integration of arbitrary domain-specific knowledge (such as dictionaries or extracted data formats) and different data field recognition methods (from domain-specific heuristics to general NLP methods such as named entity recognition). We demonstrate the method application to different target domains in section 9.
- The method is independent on the format of the input documents. We use the HTML and PDF documents as the most important information source but any other document type where the styled text is available may be used as well.

We also demonstrate that our information extraction method may be integrated with DBPedia in two ways: (1) DBPedia may be used for the recognition of candidate data fields in the extracted records and (2) the extracted records may contain new data that may be linked back to existing DBPedia resources. This allows integrating new web sources to DBPedia.

2 Related Work

Information extraction from web documents is a research area that is interesting for different applications. The most important application areas include extracting data results from query result pages [1, 8, 9, 12–15] (obtained either from general search engines or specialized ones such as product search) or obtaining structured data buried in large sets of web documents [5, 10].

When considering the recently published approaches, we may identify two basic groups from the perspective of the used representation of the input document: (1) code-based approaches that use a representation of the input document code (mainly DOM for HTML documents) [6, 9, 10, 15] and (2) vision-based approaches that use some kind of visual representation of the rendered page that may be obtained by adding some visual features to the document code model [1, 8] or by using a standard page segmentation algorithm [13, 14]. However, regardless of the used document representation, all the mentioned approaches expect HTML documents at the input.

Most existing methods are based on a top-down approach which is basically presentation-driven. After creating the document model as mentioned above, the model is usually preprocessed in order to filter the content blocks regarded as noise or to locate the most probable regions of interest (called a result section [12], data sections [14] or data region [8]). Then, the individual data records are identified based on the detection of repeating structures in the model by frequency measures [9] or visual pattern detection [1, 12, 14]. The structure of the extracted information is inferred from the discovered records while using additional information such as explicit labels present in the page [1, 12, 14, 15] or even the query interface in case of the query result extraction [12, 13]. This presentation-driven approach is suitable for many applications such as the deep web crawling. On the other hand, in case of information extraction from web sources for the semantic web, structured databases or particular applications, the structure of the extracted information is typically available in advance (for example as a domain ontology) and the task is to locate the corresponding data records in the input documents.

We have identified only a few approaches that are based on a previously known ontological model of the information being extracted. The classical work by Embley et al. [6] uses a conceptual domain model that defines the lexical and non-lexical classes and the relationships among them. However, before the conceptual model may be used for information extraction, a complex input document preprocessing is required that does not take the into account the domain model and it is based on heuristics tightly related to the HTML language constructions. Similarly, our earlier work [2] uses complex vision-based document preprocessing for creating a logical model of the processed document in a form that can be later matched with an ontology-based domain model.

Our approach we present in this paper shares many ideas regarding the ontological specification of the target domain with the work of Embley et al. [6]. However, instead of a complicated document preprocessing that presents a potential point of failure, we attempt to use the ontological specification as early as possible. As we mention in the introduction, our approach proceeds in a bottom-up manner leaving the presentation style analysis to later stages. This allows to avoid the complex document preprocessing that is usually HTML-specific and it presents a potential source of errors.

We have successfully tested some of the presented concepts during the SemPub 2015 challenge [5]. Our solution [11] was however tailored to a given particular application. In this paper, we present a new method based on a general model of the target domain.

3 Task Specification

The goal of our method is extracting information corresponding to ontological *concepts* (classes) from documents. In Fig. 1, we show a sample class (a conference paper) that is taken from a larger ontology we used for a particular information extraction task [11].

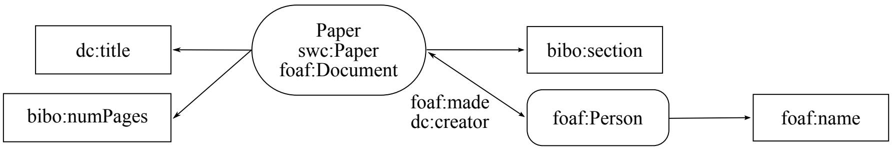


Fig. 1. Sample ontology representing a concept (Paper) and its data and object properties using the concepts and properties from the Bibliographic Ontology, FOAF Ontology and the Semantic Web Conference Ontology.The ovals represent the object properties and the rectangles represent its data properties.

According to the usual terminology in this area (for example [6]), the information about the instances of the given class (*individuals*) is represented by *data records* in the source documents. Each data record consists of multiple *data fields*, sometimes also called data units [13] that provide the lexical representation of some data properties (lexical properties) of the individual. The data fields are represented as text strings contained in the document text. Thus, a data record can be defined as a set of data fields that describe the same individual.

The task we investigate in this paper is to recognize all the data records in the source documents that belong to a single entity that is known in advance. Considering the Paper class in Fig. 1 as the input concept specification, the task is to recognize all the data records in the source documents that contain the information about individual papers containing their titles, author names, sections and pages.

4 Method Overview

We assume processing of web documents containing multiple data records corresponding to the same concept. The data records are presented in one or more source documents in a visually consistent way (we discuss the visual consistency in more detail in section 7). The key idea is to discover the most frequent visual presentation patterns that occur in the source documents and that are used for presenting the data records. Subsequently, the data records are extracted using the discovered patterns. The method in general does not involve any learning phase on a training set of documents. For every extraction task, it only analyses the presentation patterns in the given source document. However, a trained classifier may be used as one of the sources of the necessary background knowledge for certain application domains. We demonstrate one such application in section 9.4.

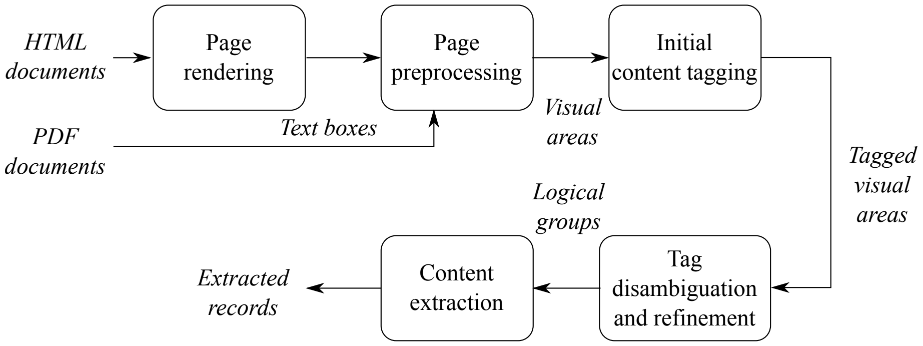


Fig. 2. Method overview

Fig. 2 shows the overview of our method. It operates on a visual representation of the source documents that is independent on the underlying code. Therefore, the first step is the document preprocessing that consists of creating an uniform representation of the source documents as a set of *visual areas.*

Next, in the initial tagging step, we perform an approximate recognition of the individual parts of the document content. This step gives a rough idea about the possible meaning of the individual visual areas; that means which visual areas might possibly correspond to some particular data fields. The result is represented by adding *tags* to the respective visual areas. Since the initial tagging is only approximate, some visual areas may obtain multiple tags and some of them may be tagged incorrectly.

Therefore, in the next step, we discover the most frequent presentation patterns used in the source documents and we use them to disambiguate and refine the assigned tags. The most supported visual patterns are then used for recognizing the desired data records.

In the following sections, we discuss the details of all the individual steps.

5 Input Document Preprocessing and Representation

The purpose of the input document preprocessing is to create a unified, format-independent model of the document content and its visual presentation. This step is typical for all the visually oriented information extraction approaches; we may mention the Visual block model used in [1], Page layout model [2] or the Visual block tree in [13].

Typically, all these models have a hierarchical structure which corresponds to the typical visual organization of the content in a web page. However, in our approach, we do not take into account the overall visual organization of the page such as visually separated block or sections. Instead, we employ a bottom-up approach, that considers only the individual parts of the text content, their visual style and mutual visually expressed relationships. Therefore, we do not need to represent the complete visual hierarchy of the page and we use only a simplified flat model consisting of a set of *visual areas* as we define below.

The input of the preprocessing step is a set of *text boxes* contained in the source document. With a text box, we understand a rectangular area in the displayed page with a know position, size containing a portion of the document text. For HTML documents, the information about the text boxes is available from a rendering engine after the document has been rendered. In case of PDF documents, this information is directly available in the source document. In both cases, the information about the visual style of the contained text (such as the used font or color) is also available for each box.

In the preprocessing step, we create visual areas from the text boxes. A visual area provides an abstraction over the rendered boxes. It is a rectangular area in the rendered page that corresponds to one or more displayed text boxes depending on the chosen *granularity* as we explain below. We define a visual area a as follows:

$$a = (rect, text, style, B) \tag{1}$$

where $rect = (x, y, w, h)$ is a rectangle representing the area position and size in the rendered page, $text$ is the text string contained in the area and $style$ is the area style:

$$style = (fs, w, st, c, bc) \tag{2}$$

where fs is the average font size used in the visual area, $w \in [0, 1]$ is the average font weight where 1 means the whole area written in bold font and 0 means the whole area written in regular font, $st \in [0, 1]$ is the average font style (1 for italic font, 0 for regular font) and c and bc are the foreground and background colors used in the area. Finally, $B = b_1, b_2, \ldots, b_n$ is the set of boxes contained in the area $(n > 0)$.

As the result of the preprocessing step, we obtain a set A of all the visual areas in the page:

$$A = a_1, a_2, \ldots, a_m \tag{3}$$

where m is the total number of visual areas in the page. Then, for any pair of visual areas $a_i, a_j \in A$ the corresponding sets of boxes B_i, B_j are disjoint $(B_i \cap B_j = \oslash)$ and the corresponding rectangles $rect_i$, $rect_j$ do not overlap.

5.1 Visual Area Granularity

The granularity of the visual areas generally depends on the application domain. In section 9, we give several examples of the application domains and we discuss the chosen visual area granularity for each of them. The highest possible granularity is obtained when for each visual area $a_i \in A$, the corresponding set of boxes B_i contains a single box ($|B_i| = 1$). However, for most application, we choose a higher granularity ($|B_i| \geq 1$). Typical choices are the following:

- *Inline-level granularity* – the visual areas are formed by sets of neighboring boxes (based on their positions on the page) that are vertically aligned to a single line and they share a consistent visual style as defined in (2). This level approximately corresponds to inline-level elements used in HTML documents.
- *Block-level granularity* – the visual areas are formed by sets of boxes that form a visually separated block of text in the page (for example a text paragraph). We use a simple block detection method proposed in [3] that is based on the discovery of clusters of adjacent boxes based on their positions in the page.

Depending on the chosen granularity, we obtain a larger or smaller set A of visual areas that represent the elementary pieces of the document content in the following steps of information extraction.

6 Initial Content Tagging

The purpose of the initial content tagging is to recognize all the visual areas that possibly might correspond to an extracted data field. Shortly, we want to identify the pieces of information that possibly "look like" some data field (for example a paper author name) when viewed separately. Each visual area is considered separately and it is assigned *tags* that indicate its possible meanings.

Based on the target domain, we define a set $T = t_1, t_2, \ldots, t_n$ of tags that may be assigned to visual areas. Each tag is identified by its name and it represents a particular data field to be extracted. For example, for the domain of conference papers shown in Fig. 1, we obtain the following set of tags corresponding to the data properties of the papers:

$$T = \{\text{title}, \text{authors}, \text{section}, \text{pages}\} \tag{4}$$

where the individual tags denote the paper title, author names, section title and page numbers respectively. For each tag, we define a *tagging* function that assigns a *support* to every visual area and tag:

$$\mathit{tagging} : A \times T \to \mathbb{R}_{[0,1]} \tag{5}$$

For a visual area $a \in A$ and a tag $t \in T$, the assigned support is a number $s \in [0, 1]$ that represents the probability that the visual area has the meaning that corresponds to the given tag. When $s > 0$, we say that the tag t has been assigned to a with the given support; for $s = 0$, we say that t has not been assigned to a. Multiple tags may be assigned to a single area (for example, the number *"15"* may be recognized as both hour and minutes in the time domain).

The initial tagging represents a highly approximate estimation of the meaning of the individual visual areas which is used as a starting point for further refining. We note that some of the tags (such as *title* and *section*) cannot be reliably distinguished when considering the visual areas separately. In that case, the visual area may obtain both tags (that means it may correspond to both the paper and section title) and later, the tags are disambiguated using the presentation context as described in section 7.

From the practical point of view, we implement the *tagging* function as a set of *taggers* where the tagger is a procedure that is responsible for computing the support of a single particular tag given a visual area. The tagger implementation may be very variable but generally, we consider the following approaches to the tag assignment that may be combined arbitrarily:

- *DBPedia concept annotation* for example using the *Spotlight* tool [4].
- *Named entity recognition (NER)* may be used for recognizing the entities such as personal names or locations depending on the used NER classifier.
- Occurrences of *keywords* (for example month names), *numerical values* in given ranges or specific *regular expressions.*
- *Visual classification.* As we have shown in our earlier work [3], it is possible to use the visual features of the areas such as the used font, colors, position within the page or amount of contained text to create a classifier, that is first trained on a set of manually annotated documents and then, it may be used for recognizing the meaning of new, previously unseen visual areas in new documents. Unlike the remaining tagging methods, the visual classification approach requires a training set of documents for setting up the classifier as we show on a practical example in section 9.4. However, the trained classifier may be later used for a whole set of documents comming even from different web sources.

For each tag, there is a single tagger defined that takes into account different criteria. The tagger may combine multiple methods with different supports. For example, the personal names may be recognized by DBPedia concept annotation (with the highest support) but the NER classifier may be used as a fallback solution (with a lower support) for recognizing the names that have no corresponding DBPedia resource.

7 Tag Disambiguation

After the visual areas have been approximately tagged, we disambiguate the tags by considering combinations of the data fields that are expected in the extracted data records (for example considering the *title* – *authors* or *title* – *pages* combination in our example in Fig. 1). We assume that all the data records are presented in a visually consistent way in the source document. Based on this assumption, we define presentation constraints on the data records that must apply for considering the records to be visually consistent. Then, the disambiguation task consists of finding the best matching record presentation and layout that meets the visual consistency constraints on one side and covers as many tagged visual areas as possible on the other side.

7.1 Visual Presentation Constraints

For considering the data records to be visually consistent, we require both the consistent presentation style of the individual data fields and consistent layout of the individual fields that form a single data record.

Text Style Consistency. For the individual text fields, we require that the visual areas with the same tag assigned (for example all the paper titles) have the same visual style in the document. We have defined the area style as a tuple of visual features (2). Let's consider a set of set of visual areas A_t that have the tag t assigned and let S_t be a set of styles of all the visual areas in A_t. Then, let n_f be the number of visual features that have equal values for all the styles $s \in S_t$. We say that A_t has a consistent style if n_f is over certain threshold.

Based on our practical experiments, we allow one visual feature that is often used by the document authors to further distinguish the individual records (for example some papers considered to be more important have a bold title or use a different color). Therefore, we use $n_f = 4$ for our experiments.

Content Layout Consistency. The layout consistency constraint is based on our assumption that the layout relationships between the individual data fields expressed by their mutual positions in the page are the same for all data records. For this purpose, we define four relations $R_{side}, R_{after}, R_{below}, R_{under} \subseteq A \times A$ that are defined based on the positions of the areas in the page. Considering a pair of visual areas $a_1, a_2 \in A$ and their respective positions $rect_1, rect_2$ in the page, we define the relations as follows:

- $(a_1, a_2) \in R_{side}$ when a_1 and a_2 are on the same line (their y coordinates overlap), a_2 is placed to the right of a_1 without any other visual area being placed between a_1 and a_2 and the horizontal distance between a_1 and a_2 is not larger than 1 em[1] (shortly, a_2 placed next to a_1).
- $(a_1, a_2) \in R_{after}$ when a_1 and a_2 are on the same line and a_2 is placed to the right of a_1 anywhere on the line (a_2 is on the same line after a_1).
- $(a_1, a_2) \in R_{under}$ when a_1 and a_2 are placed roughly in the same column (their x coordinates overlap) and a_2 is placed below a_1 without any other visual area being placed between a_1 and a_2 and the vertical distance between a_1 and a_2 is not larger than 0.8 em (a_2 is placed just below a_1).
- $(a_1, a_2) \in R_{below}$ when a_1 and a_2 are placed roughly in the same column (their x coordinates overlap) and a_2 is placed anywhere below a_1.

As we may notice, $R_{side} \subseteq R_{after}$ and $R_{under} \subseteq R_{below}$. For each pair of data fields, we choose the most supported one by trying to cover as many tagged

[1] In typography, 1 em is a length corresponding to the point size of the current font.

visual areas as possible using each relation. Since one-to-many relationships are allowed between the data fields, any of the above relations may turn out to be the most supported one.

7.2 Matching the Visual and Semantic Relationships

The tag disambiguation in our approach is based on discovering the most supported combinations of the tagged areas in the page. Considering the target domain described by an ontology (such as our example in Fig. 1), we find the binary relationships with the one-to-many or one-to-one cardinality between the different data properties in the ontology. We assume that the same semantic relationships between two data properties are represented by the same layout relation between the corresponding visual areas for all the data records in the page and in the same time, the visual areas corresponding to the same data type properties have the consistent visual style as defined in section 7.1.

In our sample ontology, we may identify the following one-to-many (or one-to-one) relationships that are expected to have a corresponding visual representation in the document: *section – title*, *title – author*, *title – pages*. Note that the paper title may be viewed as a record-identifying field here as defined in [6].

Let's consider a single relationship between the properties represented by the tags t_1 and $t_2 \in T$. Let s_{min} be a minimal value of the tag support (5) that is required for considering the area to have the given tag assigned and let A_{t_1} and A_{t_2} be the sets of visual areas that have the respective tags assigned:

$$A_{t_1} = \{a \in A : ((a, t_1), s) \in \mathit{tagging} \wedge s \geq s_{min}\} \quad (6)$$

$$A_{t_2} = \{a \in A : ((a, t_2), s) \in \mathit{tagging} \wedge s \geq s_{min}\} \quad (7)$$

and let S_{t_1} and S_{t_2} be the sets of all styles (2) of the visual areas that belong to A_{t_1} and A_{t_2} respectively. We define a *configuration* of a record extractor as follows:

$$c = (s_{t_1}, s_{t_2}, R) \quad (8)$$

where $s_{t_1} \in S_{t_1}$, $s_{t_2} \in S_{t_2}$ and R is a layout relation as defined in section 7.1. For each such configuration, we may find a set of matching pairs of visual areas:

$$M_c = \{(a_1, a_2) : a_1 \in A_{t_1} \wedge a_2 \in A_{t_2} \wedge \mathit{style}(a_1) = s_{t_1} \wedge \mathit{style}(a_2) = s_{t_2} \wedge (a_1, a_2) \in R\} \quad (9)$$

where $\mathit{style}(a_1)$ and $\mathit{style}(a_2)$ are the styles of a_1 and a_2 respectively. The goal of the tag disambiguation to find a configuration c with the largest set M_c of the corresponding area pairs.

The whole tag disambiguation algorithm for a pair of tags t_1, t_2 corresponding to a one-to-many relationship in the domain ontology may be summarized in the following steps:

1. Compute A_{t_1}, A_{t_2} and the corresponding sets of styles S_{t_1} and S_{t_2} with the minimal support s_{min} set to a higher value (we use $s_{min} = 0.6$ for considering only the tags assigned with some safe support).

2. Compute M_c for all possible configurations c and find the resulting configuration $c_x = (s_{t_1x}, s_{t_2x}, R_x)$ where M_{c_x} is the largest set of matching pairs.
3. Decrease s_{min} and recompute A_{t_1}, A_{t_2} and S_{t_1} and S_{t_2} in order to consider even the areas with the tags assigned with a low support (we use $s_{min} = 0.1$).
4. Recompute M_{c_x} for the previously discovered configuration c_x.

After the last step, M_{c_x} contains visually consistent pairs of visual areas (a_1, a_2) that correspond to the same pairs of data fields in the data records.

This process may be generalized to consider multiple one-to-many relationships: we just search for multiple configurations c while maintaining the consistency of s_{t_1} and s_{t_2} and we obtain one set M_{c_x} for each one-to-many relationship. For the one-to-one relationships, the process is equal; the only difference is in the M_c size computation where we consider all the (a_1, a_i) pairs (for all available values of i) as a single pair when computing the size of M_c.

8 Record Extraction

The obtained sets of matches M_c identify the visual areas that contain the corresponding data fields from all the data records discovered in the document. Since the visual areas are directly linked to text boxes from the source document (1), the text content contained in the area may be obtained by a simple concatenation of the text contents of the text boxes.

Depending on the target domain and the area granularity chosen in the preprocessing step (see section 5.1), it may be necessary to further postprocess the extracted text. The postprocessing includes converting the text content to particular data types (such as numbers or dates) or cleaning the text from an additional content. Finally, the obtained values may be mapped to the appropriate ontological properties.

9 Experimental Evaluation

We have implemented the proposed method of data records extraction in Java using our FITLayout framework[2]. The framework is able to process the HTML and PDF input documents by using the CSSBox rendering engine[3]. In order to demonstrate the applicability of the method, we have chosen four application domains, each having some specific features. Although it is not our primary aim to outperform the existing methods in terms of precision, we provide the evaluation of the achieved precision and recall for each sample application in order to show that the obtained results are usable in practice.

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

[3] `http://cssbox.sourcegorge.net/`

9.1 Conference Papers

For the conference paper domain, we have used a dataset from the Semantic Publishing Challenge at the ESWC 2015 conference [5]. The input dataset consists of 148 selected CEUR workshop proceedings pages[4] from the years 1994 – 2014 containing the metadata about 2,500+ papers. The input HTML documents are very variable regarding both the code and the visual style. On this dataset, we would like to demonstrate that our approach is able to automatically adapt to the presentation style used in each document and based on the specified domain knowledge, it is able to extract the paper information from a large set of diverse documents.

The extraction task is defined by ontology in Fig. 1 and a set of taggers for assigning the *title*, *authors*, *section* and *pages* tags. For tagging the possible *authors*, we have used the Stanford NER classifier [7] for personal name recognition. The remaining taggers are implemented using regular expressions defining the allowed format of the corresponding data fields.

Table 1. Results for the conference papers task: number of records extracted, precision, recall and F-measure for two different data sets.

Data set	**#rec**	**P**	**R**	**F**
(A) Complete dataset (115 documents)	2420	0.976	0.955	0.966
(B) Only documents containing page numbers	883	0.997	0.975	0.986

Since not all the documents contain the page numbers and sections, we have run two experiments: (A) on the complete data set (148 documents) with matching only the *title* – *authors* pairs and (B) on a subset of documents containing the sections and page numbers (67 documents) with matching the complete records. We have used the evaluation data provided by the SemPub Challenge organizers to evaluate our results and we provide the obtained results in Table 1. As we may notice, we have obtained better results for the (B) dataset which has two main reasons: first, the (B) dataset contains newer documents that are more visually consistent and second, by adding *pages* and *section* tags, the disambiguation is more efficient (more inconsistent combinations may be excluded from the result).

9.2 Sports Results

For the demonstration of the DBPedia concept matching usage, we have chosen the sports results domain as an example of integrating a rapidly changing external data source with DBPedia. We have extracted the records containing athlete *name*, *country* and current *points* from the current tennis and cycling rankings available on the web.

We have used DBPedia Spotlight web service for recognizing the athlete names (the matched DBPedia resource should be instance of `dbo:Athlete`) and

[4] `http://ceur-ws.org/`

countries (instance of `dbo:Country`). Moreover, we have used Stanford NER classifier for recognizing the personal names a locations in case no corresponding resource is available in DBPedia. All visual areas containing a numeric value are considered a possible *points* value and tagged with the corresponding tag.

Table 2. The sports results tasks

Source	**#rec**	**P**	**R**	**F**
ATP rankings (tennis.com)	200	1.000	0.935	0.966
WTA rankings (tennis.com)	200	1.000	0.925	0.961
Road cycling rankings (uci.ch)	2488	1.000	0.933	0.965
Mountain bike rankings (uci.ch)	1627	1.000	0.978	0.989

For every source document[5], we have prepared the "golden standard" data for evaluation by manually transforming the source HTML code to a structured CSV table using a text editor. The results in Table 2 show that based on the assigned tags, our method is able to automatically infer the presentation pattern used for presenting the data records and extract the records with a high precision. In a few cases, the personal names are not identified correctly (there is no corresponding DBPedia resource and the NER classifier failed to recognize the name) which is the reason of lower recall. The resulting extracted records are linked to the corresponding athlete resources in DBPedia. This demonstrates the possibility of an easy integration of an external resource with DBPedia without any predefined templates.

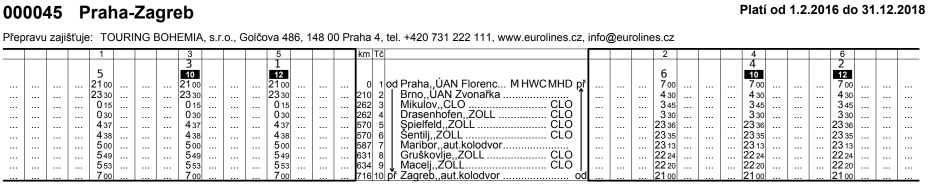
9.3 Timetables

Timetables provide a data source containing an extremely low amount of labels and other additional information that could be used for the data interpretation. Actually, a timetable often contains only the data (hours, minutes, station names) formatted in a specific way leaving its interpretation to a great extent on the experience of the human reader. Motivated by a practical need, we have used the timetables available at the official Czech public transportation timetable portal.[6] The timetables are published here in PDF files (see Fig. 3 for an example) providing a good example of processing data-rich PDF documents.

The domain knowledge is represented by an ontological description in Fig. 4 and taggers for the tags *hours* and *minutes* based on the recognition of numbers in the corresponding range and for *stops* (stop names) based on matching with a fixed list of existing stops (which is available in this domain) combined with regular expressions used when the matching fails.

[5] The URLs of the source documents were `http://www.tennis.com/rankings/ATP/`, `http://www.tennis.com/rankings/ATP/`, `http://www.uci.ch/road/ranking/` and `http://www.uci.ch/mountain-bike/ranking/` respectively.

[6] `http://portal.idos.cz`



000045 Praha-Zagreb **Platí od 1.2.2016 do 31.12.2018**

Přepravu zajišťuje: TOURING BOHEMIA, s.r.o., Golčova 486, 148 00 Praha 4, tel. +420 731 222 111, www.eurolines.cz, info@eurolines.cz

				1				3				5				km	Tč								2				4				6			
								3				1																	4				2			
				5				10				12													6				10				12			
...	...	...	...	21 00	...	...	...	21 00	...	...	...	21 00	...	...	...	0	1	od	Praha,,ÚAN Florenc...	M HWC MHD	př	...	...	...	7 00	...	...	...	7 00	...	...	...	7 00	...	...	...
...	...	...	...	23 30	...	...	...	23 30	...	...	...	23 30	...	...	...	210	2		Brno,,ÚAN Zvonařka		↑	...	...	...	4 30	...	...	...	4 30	...	...	...	4 30	...	...	...
...	...	...	...	0 15	...	...	...	0 15	...	...	...	0 15	...	...	...	262	3		Mikulov,,CLO	CLO		...	...	...	3 45	...	...	...	3 45	...	...	...	3 45	...	...	...
...	...	...	...	0 30	...	...	...	0 30	...	...	...	0 30	...	...	...	262	4		Drasenhofen,,ZOLL	CLO		...	...	...	3 30	...	...	...	3 30	...	...	...	3 30	...	...	...
...	...	...	...	4 37	...	...	...	4 37	...	...	...	4 37	...	...	...	570	5		Spielfeld,,ZOLL	CLO		...	...	...	23 36	...	...	...	23 36	...	...	...	23 36	...	...	...
...	...	...	...	4 38	...	...	...	4 38	...	...	...	4 38	...	...	...	570	6		Šentilj,,ZOLL	CLO		...	...	...	23 35	...	...	...	23 35	...	...	...	23 35	...	...	...
...	...	...	...	5 00	...	...	...	5 00	...	...	...	5 00	...	...	...	587	7		Maribor,,aut.kolodvor....................			...	...	...	23 13	...	...	...	23 13	...	...	...	23 13	...	...	...
...	...	...	...	5 49	...	...	...	5 49	...	...	...	5 49	...	...	...	631	8		Gruškovlje,,ZOLL	CLO		...	...	...	22 24	...	...	...	22 24	...	...	...	22 24	...	...	...
...	...	...	...	5 53	...	...	...	5 53	...	...	...	5 53	...	...	...	634	9	↓	Macelj,,ZOLL...........................	CLO		...	...	...	22 20	...	...	...	22 20	...	...	...	22 20	...	...	...
...	...	...	...	7 00	...	...	...	7 00	...	...	...	7 00	...	...	...	716	10	př	Zagreb,,aut.kolodvor		od	...	...	...	21 00	...	...	...	21 00	...	...	...	21 00	...	...	...

Fig. 3. A sample timetable

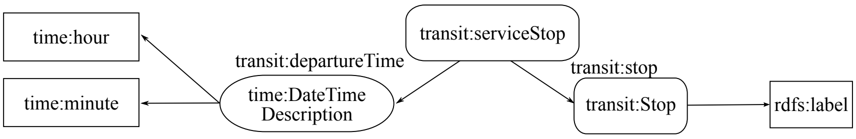


Fig. 4. Ontology used for timetables. The concepts and properties come from the OWL Time ontology and Transit ontology

We have tested our method on 30 different time table documents from the above mentioned portal. The extracted data was compared with a golden standard that was created manually by transforming the PDF files to CSV data using a semi-automatic transformation based on regular expressions tailored for the particular documents. Because the time tables contain a large amount of $(hour, minute, stop)$ records (we have obtained 5130 records in total), the tag disambiguation works very efficiently in this case and we have extracted all the records correctly $(P = R = 1.0)$. It is worth noting that all the *hour* and *minute* pairs have been identified correctly although the initial tagging is very ambiguous (many visual areas share both tags after the initial text classification).

9.4 News Articles

We have chosen the news articles domain to demonstrate a different application scenario. Unlike the documents in the previous domains that typically contained many data records (papers or times), the news web pages usually contain one full article in a document. However, each news website contains many such documents that follow a visually consistent presentation style. Therefore, we may treat a set of documents as a single input page containing multiple articles.

For this task, we view the individual news articles as data records containing data fields that we have assigned the following tags: *title* (article title), *author* (author name), *pubdate* (date of publication) and *paragraph* (a paragraph of text). Considering the title to be the record-identifying field, the $title - paragraph$ pairs correspond to a one-to-many relation, the $title - author$ and $title - pubdate$ pairs are one-to-one relations.

Due to the specific properties of the news domain where it may be difficult to recognize the individual parts such as titles and subtitles by text classification only, we employ a visual classification approach that allows to assign the tags to the areas based on their visual appearance. This approach that we have presented in detail in [3] uses the visual features of the individual visual areas: Font features such as the font size, weight and style, spatial features (position in the page and size), text features (numbers of characters and lines) and color features (luminosity, contrast). The values of the features are expressed numerically and used as an input for a generic classifier[7]. Therefore, in contrast to the other applications presented in the previous sections, a training set of documents is required for setting up the classifier. Later in the classification step, the trained classifier directly assigns tags to the visual areas in new documents.

Table 3. Results for the news articles task: precision, recall and F-measure with and without using tag disambiguation

Method	**Precision**	**Recall**	**F-measure**
Visual classifier only	0.593	0.790	0.678
Visual classifier + disambiguation	0.978	0.986	0.982

For testing, we have used the news articles on reuters.com and cnn.com news portals. We have taken 30 documents with articles from each website. We have manually annotated the source documents by manually assigning the appropriate tags to the individual visual areas in the documents.[8] Then, 5 documents from each web site were used for training the classifiers (one for each source website) based on the visual features of the manually tagged areas. Later, the trained classifiers were used for assigning tags to all the visual areas in the complete dataset from the given website.

The results obtained are shown in Table 3. The first row shows the values obtained by comparing the classification results with the manually assigned tags. This corresponds to the scenario presented in [3]. The second line shows the result with disambiguation where the visual classification results were used as the initial tagging for the tag disambiguation process described in section 7. As we may see, the disambiguation greatly improves the resulting precision and recall.

10 Conclusions

We have presented a record extraction approach from web documents that is based on searching the most frequent visual presentation patterns in the documents while assuming that multiple instances of the records are available in the documents. The extraction itself is based only on the knowledge available for the target domain that includes the expected structure of the extracted records and

[7] For our experiments, we have used the J.48 classifier from the WEKA package (which is an implementation of the C4.5 decision tree classifier) mainly for its speed.

[8] The used FITLayout framework provides a graphical annotation tool that was used for this task.

an estimation of possible values (or alternatively a style) of the data fields. We consider this as the main benefit of the presented approach. As the result, the method is independent on the source document format, and it does not rely on any kind of templates used or labels or annotations present in the source documents. The experimental results demonstrate the applicability of the approach for different scenarios and document sources.

Acknowledgments

This work was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II); project IT4Innovations excellence in science – LQ1602.

References

1. Anderson, N., Hong, J.: Visually extracting data records from query result pages. In: Web Technologies and Applications: 15th Asia-Pacific Web Conference, APWeb 2013, Sydney, Australia, April 4-6, 2013. Proceedings. pp. 392–403. Springer, Berlin, Heidelberg (2013)
2. Burget, R.: Hierarchies in HTML documents: Linking text to concepts. In: 15th International Workshop on Database and Expert Systems Applications. pp. 186–190. IEEE Computer Society (2004)
3. Burget, R., Burgetová, I.: Automatic annotation of online articles based on visual feature classification. International Journal of Intelligent Information and Database System 5(4), 338–360 (2011)
4. Daiber, J., Jakob, M., Hokamp, C., Mendes, P.N.: Improving efficiency and accuracy in multilingual entity extraction. In: Proceedings of the 9th International Conference on Semantic Systems (I-Semantics) (2013)
5. Di Iorio, A., Lange, C., Dimou, A., Vahdati, S.: Semantic publishing challenge–assessing the quality of scientific output by information extraction and interlinking. In: Semantic Web Evaluation Challenges, pp. 65–80. Springer (2015)
6. Embley, D.W., Campbell, D.M., Jiang, Y.S., Liddle, S.W., Lonsdale, D.W., Ng, Y.K., Smith, R.D.: Conceptual-model-based data extraction from multiple-record web pages. Data Knowl. Eng. 31(3), 227–251 (Nov 1999)
7. 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)
8. Goh, P.L., Hong, J.L., Tan, E.X., Goh, W.W.: Region based data extraction. In: Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on. pp. 1196–1200 (May 2012)
9. 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`
10. 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

11. Milička, M., Burget, R.: Information extraction from web sources based on multi-aspect content analysis. In: Semantic Web Evaluation Challenges, SemWebEval 2015 at ESWC 2015. Communications in Computer and Information Science, vol. 2015, pp. 81–92. Springer International Publishing (2015)
12. Su, W., Wang, J., Lochovsky, F.H.: Ode: Ontology-assisted data extraction. ACM Trans. Database Syst. 34(2), 12:1–12:35 (Jul 2009), http://doi.acm.org/10.1145/1538909.1538914
13. Weng, D., Hong, J., Bell, D.A.: Extracting data records from query result pages based on visual features. In: Advances in Databases: 28th British National Conference on Databases, BNCOD 28, Manchester, UK, July 12-14, 2011, Revised Selected Papers. pp. 140–153. Springer, Berlin, Heidelberg (2011), http://dx.doi.org/10.1007/978-3-642-24577-0_16
14. Weng, D., Hong, J., Bell, D.A.: Automatically annotating structured web data using a svm-based multiclass classifier. In: Web Information Systems Engineering – WISE 2014: 15th International Conference, Thessaloniki, Greece, October 12-14, 2014, Proceedings, Part I. pp. 115–124. Springer International Publishing, Cham (2014), http://dx.doi.org/10.1007/978-3-319-11749-2_9
15. Zheng, X., Gu, Y., Li, Y.: Data extraction from web pages based on structural-semantic entropy. In: Proceedings of the 21st International Conference on World Wide Web. pp. 93–102. WWW '12 Companion, ACM, New York, NY, USA (2012), http://doi.acm.org/10.1145/2187980.2187991