Association for Information Systems
AIS Electronic Library (AISeL)

ECIS 2013 Completed Research | ECIS 2013 Proceedings

7-1-2013

Deepec: An Approach For Deep Web Content Extraction And Cataloguing

Augusto F. Souza
Universidade Federal de Santa Catarina, Florianópolis, SC, Brasil, augustofs@gmail.com

Ronaldo S. Mello
Universidade Federal de Santa Catarina, Florianópolis, Brazil, ronaldo@inf.ufsc.br

DEEPEC: AN APPROACH FOR DEEP WEB CONTENT EXTRACTION AND CATALOGUING

Souza, Augusto Ferreira de, University Federal of Santa Catarina, Florianópolis, Brazil, augustofs@gmail.com

Mello, Ronaldo dos Santos, University Federal of Santa Catarina, Florianópolis, Brazil, ronaldo@inf.ufsc.br

Abstract

This paper presents DeepEC (Deep Web Extraction and Cataloguing Process), a new method for content extraction of Deep Web databases and its subsequent cataloguing. Our focus is on the extraction of hidden Web content presented in HTML pages generated from Web forms query submissions. While state-of-the-art information extraction and cataloguing methods address this issue separately, DeepEC is able to simultaneously perform the extraction and cataloguing of data without expert user intervention. This is accomplished with the support of a knowledge base that allows semantic inference for relevant records to be extracted and then catalogued. An experimental evaluation on some Deep Web domains shows that DeepEC achieves very good results. If compared to related work, DeepEC provides a unified process for Deep Web content extraction and cataloguing, being able to infer missing values for extracted records to be catalogued.

Keywords: Deep Web, data extraction, data cataloguing, knowledge base.

1 INTRODUCTION

The *Deep Web* refers to the content on the World Wide Web which is not indexed by standard search engines and comprises, among other information, all "hidden" databases behind Web applications, like car dealers, hotel booking and e-commerce (Halevy et al. 2009). The increasing volume of data available on the Deep Web increases, in turn, the need for accessing this kind of information by users. However, schema and data of Deep Web databases are hard to discovery as well as to extract because their structure and content become visible (or partially visible) only when shown on dynamic pages created as a result of a query specified over a Web form (Bergman, 2001).

To get access to Deep Web information, specific solutions are required to discover Web forms that act as entry points to hidden databases, like focused crawlers, meta-searchers and integration systems. However, after the discovery process, data extraction techniques are needed to allow users' access to these data sources. This is a challenging task because schema and data are heterogeneous and may be also formatted for presentation in many different ways.

Data extraction is a common problem for various processes related to data management, such as structure/data discovery and similarity search ((Ferrara and Fiumara, 2010) (He et al. 2003) (Meng et al. 2010) (Oro and Ruffolo, 2011)). In particular, a complete extraction process for Deep Web data corresponds to the acquisition and further cataloguing of relevant information among raw data, such as the extraction of metadata (attributes) of Web forms, as well as content extraction of query results shown in Web pages.

The motivation of this paper is the lack of efficient solutions related to the extraction of relevant information from Deep Web databases, in particular, the cataloguing of extracted information for further consumption by users of information systems. One example could be a searching for car rental Web sites based on specific values for car make, model, year and/or price.

This problematic is the focus of this paper, i.e., extraction and cataloguing of relevant content and structure presented in hidden databases on the Web. We consider that the data to be extracted comes from dynamic pages that hold query results. This is still a challenging and open problem due to the existence of a wide variety of Web sites with different patterns for showing the content of these databases and the existence of many useless information (menus, ads, etc.) that hinder the identification and extraction of relevant content.

On reviewing related work, we verify that no one still address the issues of cleaning/adjustment and cataloguing the discovered content besides data extraction. Consider, for example, in Figure 1, the result page of a query submitted to a car dealer Web site. Current extraction methods usually make the selection of relevant content (data shown in the central frame), removing menus and advertisements. However, no further data classification and cataloguing are provided in order to treat these data as useful information in one or more domains.

In order to deal with these issues, this paper presents *DeepEC*, a new method that performs relevant data extraction based on the existing work, but, in addition, it stores these data in a structured and domain-aware way with the intention to facilitate structured queries over Deep Web catalogued content. Thus, our first contribution is the simultaneous extraction and cataloguing of Deep Web data.

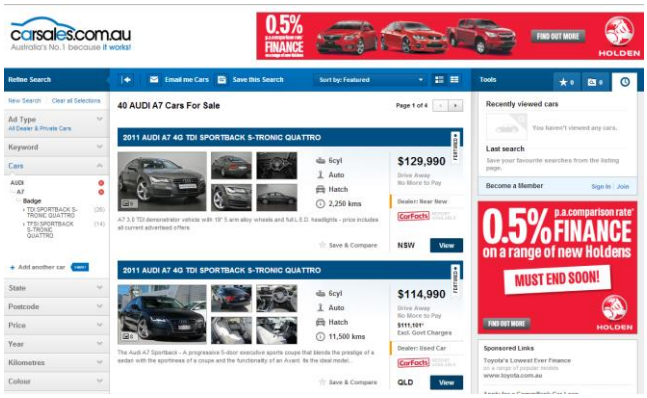

Figure 1. Example of Web page with a hidden database query result.

Table 1 illustrates the result data structuring of DeepEC execution over the Web page of Figure 1. A Knowledge Base (KB) with data from the more frequent Deep Web domains aids on the process of discovering the correct meaning of each relevant data that is extracted in order to generate a relational database with "surfaced" data from Deep Web databases.

Make	Model	Km	Price	Year	…	**Domain**
Audi	A7	2,250	129,990	2011	…	**Auto**
Audi	A7	11,500	114,990	2011	…	**Auto**

Table 1. Example of DeepEC extraction result with complemented information highlighted.

Thus, a second contribution of our method is the ability of inferring values for missing domain attributes´ contents in the result data structure. This is useful for cases where attribute values are absent in a given result page or it is not properly extracted. For example, if a record with values "civic", "automatic", "2012" is extracted, DeepEC recognizes that these data belong to the domain of automobiles (Auto) and are related to a car whose brand is "Honda".

More details about DeepEC method are given in the following. Section 2 of this paper discusses related work. Section 3 describes the method, while Section 4 presents the results of an experimental evaluation. Finally, Section 5 is dedicated to the conclusion and future work.

2 RELATED WORK

This section reviews the main features of some work in the literature related to Web data extraction and cataloguing.

2.1 Data Extraction

Web data extraction focuses on the definition of algorithms that identify and retrieve information presented on Web pages (Kaiser and Miksch, 2005). The main techniques applied to data extraction are the following:

- *Tree-based*: this technique converts the content of an HTML document into a tree structure, using mainly the standard DOM model. DOM (*Document Object Model*) is a cross-platform and language-independent convention and API for representing and interacting with objects in HTML, XHTML and XML documents. Works that adopt this approach usually search for specific hierarchical patterns in order to recognize relevant data to be extracted ((Kim et al., 2007) (Liu et al., 2003) (Zhai and Liu, 2005));

- *Web Wrappers*: this class of technique, summarized in (Ferrara and Fiumara, 2010), develops specific procedures that seek and find data required by a user, extract them from unstructured (or semistructured) Web sources, and transform them into structured data, merging and unifying this information for further processing, in a semi-automatic or fully automatic way ((Liu et al., 2000) (Muslea et al., 2001)). They are designed in three steps: *(i) generation*, which is defined according to some techniques; *(ii) execution*, that runs and extracts information continuously; and *(iii) maintenance*, considering that data sources structure may change and the wrapper must be adapted accordingly to continue working;
- *Machine Learning*: techniques that usually extract information from a specific Web source domain. They define extraction rules based on delimiters from training sessions. The training data usually comprises large amounts of manually labeled pages, requiring a lot of human involvement ((Hsu et al., 1998) (Phan et al.2005));
- *Ontology*: techniques based on the analysis of a knowledge representation of a particular domain that it is shared to various applications. Construction of this knowledge is a time consuming task done manually by experts in ontology design as well as in the application domain. For a specific application domain, the ontology is used for locating the Web page and the construction set of objects with it. The more accurate and flexible is the semantic representation, more automated is the extraction. An example of such technique is (Embley et al., 1998).

For extraction purposes, we evaluate, in this paper, two well-known data extraction approaches. One is the *Road Runner* algorithm, applied to data extraction over multiple pages (Crescenzi et al., 2001). It is a hybrid approach, because it applies both tree-based and a machine learning techniques. The other one is *MDR* (*Mining Data Records*) (Liu et al. 2003) which is a tree-based technique that traverses the structure in a depth-first way looking for patterns that represent data records.

2.2 Data Cataloguing

With respect to data cataloguing, the work of (Zhao et al., 2008) presents a technique for text segmentation based on the concept of CRFs (*Conditional Random Fields*). CRF uses tables with structured data as reference for each domain. A sample is used as input to build schemas of CRF. These schemas are based on the recognition and training of the data input sequence for each attribute in the reference table. After this training, where is defined the sequence of the attributes in the generated schema, the schema is applied to the rest of the entry for cataloguing.

ONDUX (*On-Demand Unsupervised Learning for Information Extraction*) (Cortez et al., 2010) and JUDIE (*Joint Unsupervised Structure Discovery and Information Extraction*) (Silva et al., 2011) take into account an automatic induction technique for cataloguing, i.e., features learned with data obtained from an input source are used to induce the structure-related from other distinct sources in the same domain. Given as input a text containing a set of data records, the approach initially segments data and labels the potential values, comparing them with a set of pre-existing data in a knowledge base. After this procedure, an algorithm, based on a model of data positioning and sequencing, labels them again, confirming or making corrections on the labels initially catalogued. The difference between the two approaches is that ONDUX requires user intervention to define each input record in the data sample, as well as to define the structure of the records to be extracted.

Although JUDIE does not take into account the attribute label in the cataloguing process, it is used as the baseline for this paper, as compared to ONDUX, because it does not require user intervention to delimit the records in the text input. It is important to observe that the record structure plays a significant role in this context, since the extraction task depends on particular features (positioning and sequencing) to work properly.

In general, the limitations of these works are the fact they do not consider metadata extraction from Deep Web search result pages (attribute extraction) and its associated values, as well as a process for cataloguing the extracted data in order to facilitate future reference, e.g., to assist future extractions of

pages in the same domain based on the features of this domain. Another drawback is the absence of index structures for Deep Web sites based on the terms extracted from the content of the hidden databases. These limitations had motivated the design and implementation of DeepEC, which is detailed in the following.

3 DeepEC Overview

In this section, we present our method by describing the two major modules that comprise it. Figure 2 presents the DeepEC architecture. It supports extraction and cataloguing of hidden databases relevant content from HTML pages obtained through the submission of queries formulated over Web forms.

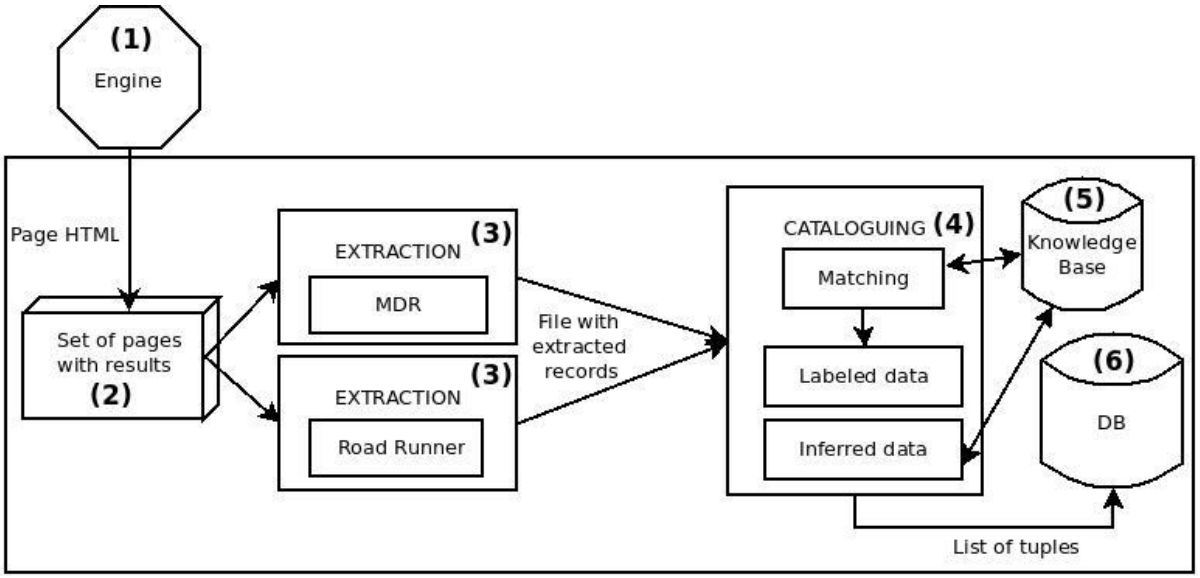

Figure 2: DeepEC architecture.

The input for DeepEC is generated by an external component called the *Engine* (1). It encapsulates the discovery process of hidden databases on the Web followed by Web form filling out, query submission and generation of HTML pages with the result set (2). The two main components that comprise our method are *Extraction* (3) and *Cataloguing* (4) modules, and, besides that, DeepEC also interacts with two data repositories: a *Knowledge Base* (5) and a relational database (*DB*) (6). All of these modules and repositories are described in the following.

3.1 Extraction

Given an input HTML page containing a set of implicit structured data records from a hidden Web database, such as the one illustrated in Figure 1, the Extraction modules outputs a set of relevant records. As mentioned before, we are currently considering two extraction algorithms with good performance: Road Runner and MDR. Both are highly referenced in the literature ((Hong, 2010) (Oro et al. 2011) (Zhai et al. 2005)).

Road Runner focuses on automatic generation of templates to extract data by matching features from different pages in the same domain. MDR also automatically identifies and extracts data records with regular structures. Our intention in this paper is also to compare their performance in terms of hidden Web database record extraction, as described in Section 4. We limit to compare only this two well-know extraction approaches, because we intend to develop, as a future work, our own approach based on the best features of them. We will detail them in the following.

3.1.1 MDR

MDR algorithm initially assembles the input HTML page in a DOM tree (Liu et al. 2003). Figure 3 shows an example of a DOM tree, where HTML tags are internal nodes and text are in the leaf nodes. Note that the analysis of Web pages in HTML aims at discovering the information found within the leaf nodes in the DOM tree.

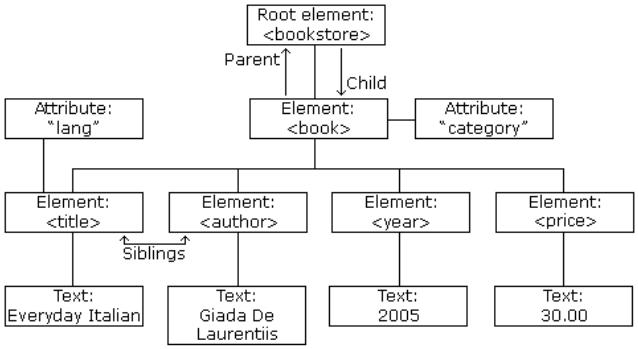

Figure 3: Example of DOM tree for an HTML page.

The code responsible for parser construction and definition of root node in MDR is presented in high level Algorithm 1. Line 3 builts the DOM structure from the HTML page. Line 4 identifies root node and line 5 calls the procedure “MDRExtract”.

Algorithm 1 Algorithm MDR.

```
1: Input ← page; /*HTML page*/
2: begin
3: PDOM ← DOMParser(page);
4: root ← DOMNode;
5: recordSet ← MDRExtract(root);
6: return ← recordSet
7: end
```

The code of the “MDRExtract” procedure is presented in the high level Algorithm 2. The algorithm traverses the tag tree from the root downward in a depth-first way (lines 3 and 4). At each internal node, procedure “CombComp” receives as parameter "childNode" (root node of the HTML document), "k" (maximum number of nodes that should be gathered for comparison), and "t" (similarity threshold to be considered in the comparisons). It performs string comparisons of various combinations of children sub-trees. Comparisons are performed by combining nodes in groups of 1 to “k” nodes and only data regions with a score equal or higher than "t" that probably represents data records are selected. To compare the trees it is used the STM algorithm (*Simple Tree Matching*) (Yang, 1991) to measure the similarity degree between two trees.

Algorithm 2 Algorithm MDR Extract.

```
1: Input: node ← DOMNode;
2: begin
3: for each childNode ∈ node.Children() do
4:     recordSet ← CombComp(childNode, k, t)
5:     return ← recordSet
6: end for
7: end
```

3.1.2 Road Runner

Given a sample with similar Web pages containing one or more data records, the Road Runner algorithm compares the HTML content of the pages to detect similarities and differences in order to create a regular expression during this analysis. At each new comparison, the expression is refined/widespread, solving differences as follows:

- Differences between text, that denotes data fields;
- Differences between tags, that denotes optional items or lists.

After the end of the comparisons, the regular expression is used to extract data records from other Web pages. The set of the extracted data is produced in HTML format.

Figure 4 shows an example of the extraction process followed by Road Runner. The extraction method works on two objects at a time: *(i)* a list of HTML pages, called the *sample*, and *(ii)* a *wrapper*, i.e., a union-free regular expression. Given two HTML pages (called page 1 and page 2), one of them is considered the initial version of the wrapper. Then, the wrapper is progressively refined trying to find a common regular expression for the two pages. This is done by solving mismatches between the wrapper and the sample.

A mismatch happens when some token in the sample does not comply to the grammar specified by the wrapper. Mismatches are very important, since they help to discover essential information about the wrapper. Whenever one mismatch is found, Road Runner attempts to solve it by generalizing the wrapper. The algorithm succeeds if a common wrapper can be generated by solving all mismatches founded during the parsing. Once the mismatch has been solved, the parsing can be resumed. In the running example of Figure 4, after solving all the mismatches, the parsing is successfully completed, generating a common wrapper for the input HTML pages.

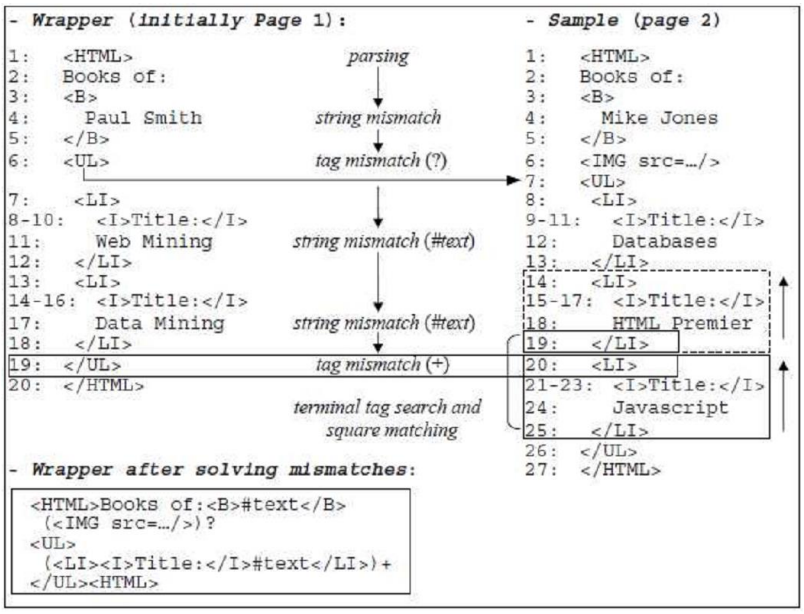

Figure4: Example of RoadRunner extraction method (Crescenzi et al., 2001).

A record set extracted by Road Runner produces a file like the one exemplified in Figure 5, for the *Auto* domain. The extraction algorithm identifies the limits of each record through the hierarchy tree and set record delimiters.

```
# Civic EX
# Gray
# 2 door
# 2012
# Coupe
# Automatic
# 18,991
# ;
# Civic Si
# Red
# 2 door
# 2012
# Manual
# Coupe
# 18,988
# ;
```

Figure 5: Example of file with extracted hidden database records.

3.2 Cataloguing

The Cataloguing module (4) follows a text segments' analysis and characterization (if they are attribute labels or values) approach based on a KB (5) built from data samples collected from several hidden database domains. This step takes as input the generated file in the previous step with the selected records (an example is shown in Figure 6), and outputs a set of tuples with cataloguing data.

```
Civic EX, Gray, 2 door, 2012, Coupe, Automatic, 18.991;
Civic Si, Red, 2 door, 2012, Manual, Coupe,  18.988;
Thomas Jefferson: The Art of Power, Random House, Jon Meacham, 18.98;
Team of Rivals: The Political Genius of Abraham Lincoln, Simon & Schuster, Doris Kearns Goodwin,11.70;
```

Figure 6: Input list with selected data

The cataloguing code is presented in Algorithm 3. In line 5, it analyses each term of the KB. For each item of each record list (lines 6 and 7) it tries to detect the domain to which the item belongs (lines 8 and 9). In line 11, we have the analysis of the specific types of data. For example, in *Auto* and *Book* domains we have specific attributes for *price* and *year*. This procedure verifies if these specific data types to these domains occur, and tries to match to the current item. The matches between KB terms and record items are detected (line 12) using comparison by similarity. Once a match is identified, the term and its definition are stored (line 13), generating a tuple to be added to the database, as shown in Table 2 for data from Figure 6.

Algorithm 3 Cataloguing Algorithm

```
 1: Input ← KB;
 2: Input ← extracted records file F;
 3: begin
 4: domain ← NULL;
 5: for each term t ∈ KB do
 6:     for each record r ∈ F do
 7:         for each item i ∈ r do
 8:             if notDetected(domain) then
 9:                 domain ← detectDomain(t, i)
10:             end if
11:             detectSpecificDataTypes(i, domain)
12:             if matching(t, i) then
13:                 catalog(i)
14:             end if
15:         end for
16:     end for
17: end for
18: end
```

One of the contributions of this work is the auto-completion of information, i.e., after identified the correspondences, we induce supplementary information that are not available on the result pages. The KB plays an important role in that context. The supplemental information is based on the KB hierarchy: if the information is not present in the list of extracted records, depending on the level that the information is stored in the KB, the method infers information from higher levels. For example, Table 2 shows the determination of the domain and make content. This information is filled out because it were not among the extracted data of Figure 6, but it was taken from KB.

Make	Model	Door	Color	Price	Year	**Domain**	URL
Honda	Civic	2	Gray	18991	2012	**Auto**	www…
Honda	Civic	2	Red	18988	2012	**Auto**	www…

Table 2: Example of Catalogued Tuple.

Our KB (5) acts as a dictionary of indexed terms. It contains attributes and sample values utilized to compare the extracted terms of the input records with the terms in the KB in order to discover their meaning. There are some free available knowledge bases and its design and usage is a current topic of research ((Chiang et al., 2012) (Serra et al., 2011)). Figure 7 shows an example of the XML structure of our KB. We had implemented wrappers for extracting data from the database of *DeepPeep* (Barbosa et al., 2010), Freebase[1] and Wikipedia[2] that were used to populate our KB.

```
<?xml version="1.0"?>
- <list>
  - <thesaurus.domain>
    - <thesaurus.auto>
        <make>HONDA</make>
        <model>CIVIC</model>
      - <thesaurus.acessory>
          <item>automatic</item>
          <item>manual</item>
          ...
        </thesaurus.acessory>
      </thesaurus.auto>
    </thesaurus.domain>
</list>
```

Figure 7: Example of DeepEC knowledge base.

In order to complete the process, the data records, represented by the labels and their values, are stored in a relational database (6), in the format described in Table 2.

4 EXPERIMENTAL EVALUATION

In this section, we describe preliminary experiments for evaluating DeepEC. We consider data from two Deep Web domains and use the traditional precision, recall and F-measure (Yates and Neto, 1999) information retrieval metrics to evaluate the tests, as presented in Figure 8. According to Figure 8, *Bi* corresponds to the set of terms that compose the values of a particular attribute in a Web page, and *Si* means the set of terms assigned by our system.

A first data set (Data set 1) used in the experiments as well as the data sources used to create the KB comes from hidden databases data for *Book* and *Auto* domains. It was considered 35 pages that create

[1] *http://www.freebase.com*
[2] *http://www.wikipedia.org/*

198 records for the Auto domain and 126 records for Book domain. This data set was used as input to DeepEC, generating the results presented in Table 3.

$$P_i = \frac{|B_i \cap S_i|}{|S_i|}, R_i = \frac{|B_i \cap S_i|}{|B_i|} \text{ and } F_i = \frac{2(R_i \times P_i)}{(R_i + P_i)}$$

Figure 8: *Precision* (P_i), *Recall* (R_i) *and F-measure* (F_i).

Table 3 shows the results organized in three columns: (i) *Extraction*: records extracted from the original Web pages; (ii) *Cataloguing*: records catalogued from the input set generated by the Extraction module; and iii) *DeepEC*: the overall approach performance. DeepEC denotes how many records were correctly catalogued based on the number of records available in the original HTML files, without considering the amount of records extracted in the extraction step.

		Extraction			**Cataloguing**			**DeepEC**		
Algorithm	**Domain**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
Road Runner	Auto	96.10	100	98.01	98.70	99.10	98.90	99.20	99.50	99.35
MDR	Auto	94.50	100	97.17	99.10	98.90	99	99.80	99.40	99.60
Road Runner	Book	95.40	99.60	97.45	97.40	98.60	98	98.30	99.10	98.70
MDR	Book	96.60	99.40	97.98	97.90	98.20	98.05	98.70	98.80	98.75

Table 3: *Results for Data Set 1.*

The results for Cataloguing module tends to present better performance because it operates on data previously extracted and well structured. Instead, Extraction module operates on very heterogeneous data that can be loosely structured. It compromises its accuracy. The results for Cataloguing module are highly influenced by the existing knowledge in the KB. On considering the chosen data set, we had obtained excellent results, as expected, since the KB was built based on data from the domains which the data set belongs. On concerning the performance of the complete system (DeepEC), experiments show excellent results regarding the amount of original records on the Web pages versus the amount of catalogued records.

In order to verify the effectiveness of our method in a deep way, we create another data set (Data Set 2) for the same domains, being these data set not included in the KB. This data set was built with 20 pages, which defines 240 records. The results are showed in Table 4. DeepEC continues to present good results on considering a complete automatic process.

		Extraction			**Cataloguing**			**DeepEC**		
Algorithm	**Domain**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
Road Runner	Auto	97.50	99.50	98.49	88.90	89.70	89.30	92.50	95.60	94.02
MDR	Auto	98.40	99.90	99.14	90.20	89.90	90.05	94.30	94.80	94.55
Road Runner	Book	98.80	98.90	98.85	92.50	90.10	91.28	96	93.90	94.94
MDR	Book	99	99	99	91.80	92.80	92.30	95.80	94.50	95.15

Table 4: *Results for Data Set 2.*

Regarding the filling out of missing information for the discovered data records, Table 5 shows the number of records that were successfully completed for both data sets. For example, in the first table line, the total number of input records was 198, but DeepEC were able to infer another 20 records that were not present in the input pages generating a gain of 10% in terms of information enrichment in the database with catalogued data.

Algorithm	Domain	Dataset	Original Records	Inferred Records	Gain (%)
Road Runner	Auto	1	198	20	10.10
MDR	Auto	1	198	14	7.07
Road Runner	Book	1	126	12	9.52
MDR	Book	1	126	12	9.52
Road Runner	Auto	2	129	12	9.30
MDR	Auto	2	129	13	10.08
Road Runner	Book	2	111	11	9.91
MDR	Book	2	111	11	9.91

Table 5: Gain with information auto-completion.

5 CONCLUSION

This paper presents DeepEC, an approach for extraction and cataloguing of Deep Web content. The motivation for DeepEC is that other approaches do not treat these tasks as a single process, presenting limitations in terms of the extraction of metadata values, and do not provide complementary information for data cataloguing purposes. DeepEC is able to perform the entire process without any human intervention, including auto-completion information during the cataloguing process. This is possible due to the support of a KB that is used for comparison and semantic inference for the extracted records.

In a preliminary experimental evaluation, we investigate two classical algorithms for data extraction and defined a KB for two Deep Web domains. For the considered domains, we note that both extraction algorithms had presented good accuracy, being hard to assert which one is the best one. In general, results show that our method had reached a very good performance while providing a gain of up to 10% in enrichment of the catalog database created for storing the extracted data.

This paper evaluates only the accuracy of the extraction and cataloguing processes. As a future work, we intend to evaluate the performance of the overall method in terms of processing time. Besides, we intend to apply DeepEC in other Deep Web domains, as well as to provide a dynamic KB, i.e., a KB that is able to learn from the extracted records and expand your knowledge. Moreover, we intend to integrate the DeepEC database with the WF-Sim database, a query-by-similarity system for Web form data developed by UFSC database group (Gonçalves et al., 2011). The idea is to improve WF-Sim query capabilities over Deep Web data including hidden database content besides Web forms content.

References

Barbosa, L., Nguyen, H., Nguyen, T. H., Pinnamaneni, R. and Freire, J.(2010). “Creating and exploring web form repositories. SIGMOD Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, p.1175-1178.

Bergman, M. K. (2001). The Deep Web: Surfacing Hidden Value. In The Journal of Electronic Publishing, Vol. 7, No. 1.

Chiang, F., Andritsos, P., Zhu, E. and Miller, R. J. (2012).AutoDict: Automated Dictionary Discovery.In IEEE 28th International Conference on Data Engineering, p. 1277-1280.

Crescenzi, V., Mecca, G. and Merialdo, P. (2001).RoadRunner: Towards Automatic Data Extraction from Large Web Sites. In Very Large Data Base (VLDB).

Embley, D. W., Campbell, D. M., Smith, R. D. (1998).Ontology-Based Extraction and Structuring of Information from Data-Rich Unstructured Documents.In CIKM Proceedings of the seventh International Conference on Information and Knowledge Management, p. 52-59.

Cortez , E., Silva, A. S., Gonçalves, M. A. and Moura, E. S. (2010). ONDUX: On-Demand Unsupervised Learning for Information Extraction. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, p. 807–818.

Ferrara, E. andFiumara, G. (2010). Web Data Extraction, Applications and Techniques: A Survey. In ACM Transactions on Computational Logic, p. 1-20.

Gonçalves, R., D'agostini, C., Silva,F. R., Dorneles, C. F., Mello, R. S. (2011). "A Similarity Search Approach for Web forms". In: Proceedings of the IADIS International Conference IADIS WWW/Internet, p. 381-387.

Halevy, A., Madhavan, J., Afanasiev, L. and Antova, L. (2009).Harnessing the Deep Web: Present and Future.In Conference on Innovative Data Systems Research (CIDR).

He, H., Meng, W., Yu, C. and Wu, Z. (2003). WISE-Integrator: An Automatic Integrator of Web Search Interfaces for E-Commerce. In Proceedings of the 29th VLDB Conference, Berlin, Germany.

Hong, J. L. (2010). "Deep Web Data Extraction". In IEEE Systems Man and Cybernetics, p. 3420 – 3427.

Hsu, C. N. and Dung, M. T. (1998).Generating Finite-State Transducers for Semi-Structured Data Extraction from the Web. In Information Systems Vol. 23, No. 8, p. 521-538.

Kaiser, K. and Miksch, S. (2005). Information extraction.Asurvey.Tech.rep., E188 – Institute of Software Technology & Interactive Systems. Vienna University of Technology

Kim, Y., Park, J., Kim, T., and Choi, J. (2007).Web information extraction by html tree edit distance matching.In International Conference on Convergence Information Technology. IEEE, p. 2455-2460.

Liu, B., Grossman, R. and Zhai, Y. (2003).Mining Data Records in Web Pages.In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.

Liu, L., Pu, C. and Han, W. (2000). XWRAP: An XML-enable Wrapper Construction System for Web Information Sources. In 16th International Conference on Data Engineering (ICDE'00), p. 611-621.

Meng, X., Liu,W. and Meng, W. (2010). ViDE: A Vision-Based Approach for Deep Web Data Extraction. In IEEE Transactions on Knowledge dnd Data Engineering, p. 447-460.

Muslea, I., Minton, S. and Knoblock, C. (2001). Hierarchical Wrapper Induction for Semistructured Information Sources. In Journal Autonomous Agents and Multi-Agent Systems archive Vol. 4 Issue 1-2, p. 93-114.

Oro, E. and Ruffolo, M. (2011). SILA: a Spatial Instance Learning Approach for Deep Web Pages. In Conference on Information and Knowledge Management (CIKM), p. 2329-2332.

Phan, X., Horiguchi, S., and Ho, T. (2005).Automated Data Extraction from the Web with Conditional Models. International Journal of Business Intelligence and Data Mining, Vol. 1, p. 194-209.

Serra, E., Cortez, E., Silva, A. S. and Moura, E. S. (2011). On Using Wikipedia to Build Knowledge Bases for Information Extraction by Text Segmentation.In Journal of Information and Data Management, Vol. 2, No. 3, p. 259–272.

Silva, A. S., Cortez, E., Oliveira, D., Moura, E. S. andLaender, A. H. F. (2011). Joint Unsupervised Structure Discovery and Information Extraction.In Special Interest Group on Management of Data (SIGMOD), p. 12-16.

Zhai, Y. and Liu, B. (2005). Web Data Extraction Based on Partial Tree Alignment. In Proceedings of the 14th international conference on World Wide Web (WWW), p. 76-85.

Zhao, C., Mahmud, J. and Ramakrishnan, I. V. (2008). Exploiting Structured Reference Data for Unsupervised Text Segmentation with Conditional Random Fields. In International Conference on Data Mining (SIAM), p. 420-431.

Yang, W. (1991) "Identifying Syntactic Differences Between Two Programs," Software Practice and Experience, vol. 21(7), p. 739-755.

Yates, R. B. and Neto, B. R.(1999)."Modern Information Retrieval.FirstEdition. Addison Wesley Longman Limited", England.