Unsupervised Wrapper Induction using Linked Data

Anna Lisa Gentile
University of Sheffield
a.l.gentile@dcs.shef.ac.uk

Ziqi Zhang
University of Sheffield
z.zhang@dcs.shef.ac.uk

Isabelle Augenstein
University of Sheffield
i.augenstein@dcs.shef.ac.uk

Fabio Ciravegna
University of Sheffield
f.ciravegna@dcs.shef.ac.uk

ABSTRACT

This work explores the usage of *Linked Data* for Web scale Information Extraction and shows encouraging results on the task of *Wrapper Induction*. We propose a simple knowledge based method which is (i) highly flexible with respect to different domains and (ii) does not require any training material, but exploits *Linked Data* as background knowledge source to build essential learning resources. The major contribution of this work is a study of how *Linked Data* - an imprecise, redundant and large-scale knowledge resource - can be used to support Web scale Information Extraction in an effective and efficient way and identify the challenges involved. We show that, for domains that are covered, *Linked Data* serve as a powerful knowledge resource for Information Extraction. Experiments on a publicly available dataset demonstrate that, under certain conditions, this simple unsupervised approach can achieve competitive results against some complex state of the art that always depends on training data.

1. INTRODUCTION

Information Extraction (IE) is the process of transforming unstructured or semi-structured textual data into structured representation that can be understood by machines. With the exponential growth of data published on the Web, IE is becoming increasingly important for Web-based knowledge acquisition tasks. One of the common techniques in Web-based IE is known as *wrappers*. In this context, a wrapper is generally a set of rules designed to extract data from a specific set of (semi-)structured documents that share structural similarities. Web pages are typical examples of such documents. A lot of websites use scripts to generate highly structured pages for different data records using consistent templates. For example, a yellow page website will use the same template to display information (e.g., name, address, cuisine) of different restaurants. Therefore, inducing wrappers for Web pages can enable extraction of similar data records in an automatic and effective fashion.

Wrapper induction [12, 16, 5, 6, 22] is the task of automatically learning wrappers using a collection of manually annotated Web pages as training data. It generally addresses extracting data from "detail" Web pages [3], which are pages corresponding to a single data record (or entity) of a certain type or *concept* (also called *vertical* in the literature) and render various attributes of that record in a human-readable form. An extensive range of work has been carried out to study wrapper induction in the past. However, the task remains challenging for several reasons. First, wrappers are typically induced based on training examples, which are manually labelled Web pages of particular websites. Creating such annotations require significant human effort and remains a bottleneck in the wrapper induction process [22, 8]. Second, wrappers are typically learnt specific to a website and largely depends on structural consistency. Porting wrappers across websites often require re-learnining [22]; and even very slight change in structures can cause wrappers to break. Although recent studies [3, 5, 6, 8] have focused on addressing these two issues, these methods still depend on manually labelled examples to train a wrapper; while in some cases [5], even more training data is required to enhance wrapper robustness.

In this work, we explore unsupervised approaches towards wrapper induction and propose a simple and efficient method that works for Web scale IE. The method is based on several hypotheses. First, it is possible to create large scale dictionaries for different attributes of a vertical in an unsupervised way and without the need for seed data. Second, given a collection of detail pages from a single website, we expect two properties of these pages. On one hand, they share structural similarity and therefore, the same attributes are often mentioned at the same or similar positions. On the other hand, we expect highly diverse attribute values and as a result, for a certain attribute that is usually at a certain position on each detail page, we expect to see diverse values across the website.

To build a method based on these hypothesis, we propose a Knowledge based approach where we (i) build pertinent dictionaries to (ii) annotate Web pages from a website and discover the structural patterns that encapsulate the target information.

To accomplish the first task we exploit *Linked Data* as background knowledge source to automatically generate dictionaries for each attribute of each vertical. *Linked Data* refers to a practice of describing and publishing structured data in terms of triples using universal vocabularies such that it become interlinked and more useful. The W3C *Link-*

ing Open Data Project [1] aims at publishing and interlinking Linked Datasets on the Web according to Linked Data principles. The scale of this *Web of Data* is constantly growing and, as of 2011, comprised 27 billion triples in 203 data sets [9] and research [10, 14] has shown *Linked Data* as a powerful entity knowledge base that contain billions of entities and descriptions of their attributes.

Once such knowledge has been collected from *Linked Data*, for each attribute-website pair, we use the attribute dictionary to annotate the detail pages from the website by simply matching the candidate values against the text nodes on the page. This process is fully unsupervised, and generates a large number of annotations together with their corresponding structural paths on the page, in the form of *xpaths*[2].

The annotations are incomplete and imprecise, but are useful to learn a high-accuracy wrapper in the next step. To do so, we rank each *xpath* by the number of different attribute values, and select the best *xpaths*, based on this ranking, to be the wrappers for the attribute and the website. The wrappers are applied to re-annotate the website to create the final annotations. Compared to the state of the art, our method is simple, generic and highly unsupervised. This assures high portability and extensibility: introducing the method across websites, verticals and attributes costs little human effort since no training data are required. Evaluated using a standard large scale dataset for wrapper induction [8], we show that the method achieves highly competitive results on many tasks, with an average accuracy of 0.80% on covered tasks.

The contribution of this work is two-fold. First, we introduce a simple, effective and highly flexible approach to extracting structured data from Web pages. Second, we investigate how *Linked Data* - an imprecise, redundant and large-scale knowledge resource - can be used to support Web scale IE in an effective and efficient way. Although *Linked Data* has been exploited to bootstrap specific IE tasks, such as relation extraction [21, 11], to the best of our knowledge this is the first work to use *Linked Data* for the task of Wrapper Induction. We discuss lessons learnt regarding the noisy nature of *Linked Data* and suggest future work in relevant research.

The paper is structured as follows. After an analysis of the related work (Section 2), we describe the proposed knowledge based solution for Wrapper Induction in Section 3. We test the method on a publicly available dataset (Section 4) and report experiment results in Section 5. We discuss the outcome of work in Section 6 and draw conclusions and future directions in Section 7.

2. STATE OF THE ART

Using Wrapper Induction to extract information from structured Web pages has been studied extensively. Early studies focused on the DOM-tree representation of Web pages and learn a template that wrap data records in HTML tags, such as [12, 15, 19].

As mentioned before, there are two major limitations of such methods. First, they require manual annotation on example pages to learn wrappers for those pages. This requires considerable human effort since examples are required for every attribute, vertical and website; while the learnt wrappers may fail on unseen attributes or websites [20]. To alleviate human effort, some unsupervised methods are proposed to firstly cluster Web pages that share similar structures (e.g., [2]), and then deduce a shared template for each cluster of Web pages. Two well-known studies in this stream is RoadRunner [4] and EXALG [1]. However, such methods do not recognise the semantics of the extracted data (i.e., attributes), but rely on human effort to identify attribute values from the extracted content.

The second limitation is that such wrappers are often very specific, and therefore are inflexible and not robust enough to cope with variations in the structures of Web pages. It is recognised that even a very slight change in the underlying structure of Web pages can cause the wrappers to break and have to be re-learnt. This is often referred to as the "wrapper breakage" problem [5, 18]. As suggested in [7], wrappers learnt without robustness considerations had an average life of 2 months, with, on average, 1 out of every 50 wrappers breaking every day. Thus, research in recent years has focused on developing robust wrapper induction approaches to address this issue. Dalvi et al. [5, 6] define a model to capture how Web pages evolve over time, and use the model to evaluate the robustness of learnt wrappers. A probabilistic tree-edit model is firstly trained using a collection of evolutions of Web pages (e.g., the IMDB page for the film The Godfather undergoes various changes across its lifetime, resulting in different versions). The model encodes the probability of each editing operation on a Web page over time, such as changing a <b> tag to an <i> tag, and inserting a new <div> and deleting a <br>. The model also allows one to compute the probability of a Web page evolving from one state to another, by aggregating the probabilities of each edit operation. Next, candidate wrappers in the form of XPaths are learnt using any state of the art wrapper induction approaches. Finally, the "robustness" of wrappers is evaluated using the learnt probabilistic model: each wrapper can be considered as a "future" snapshot of a Web page and its robustness can be formulated as the probability that the page transforms to this state. This model is further extended in the work by Parameswaran et al.[18] and Dalvi et al. [6]. A similar study on measuring robustness of wrappers by cost of edit is introduced by Gulhane et al. [7]. These methods however, require substantial data to train the probabilistic model. The model is also specifically fitted to the data and may not be ported to new domains.

Another stream of work addresses this issue by multi-view learners [16, 8, 20]. Hao et al. [8] designed a set of weak features that are general across attributes, verticals and websites, to identify a large amount of candidate attribute values that are likely to contain noise. Then, site-specific features (strong features) are derived in an unsupervised manner and exploited to boost the true values. While such methods have been shown to improve robustness of wrappers as well as reduce the amount of manual annotations for training, they still require seed Web pages to be annotated. Hao et al. [8] for instance require at least one website to be annotated for each vertical.

3. METHODOLOGY

Formally, we define the task as given a set of *concepts* of interest $C = \{c_1, \ldots, c_i\}$ with their attributes $\{a_{i,1}, \ldots, a_{i,k}\}$, and a website containing Web pages that describe entities

[1] `http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData`
[2] http://www.w3.org/TR/xpath/

of each concept W_{c_i}, annotate the attributes of each entity on the Web pages.

We divide our method into three steps. First (Section 3.1), for each attribute $a_{i,k}$ of each concept c_i, we generate a large dictionary $d_{i,k}$ of known possible values for $a_{i,k}$ by exploiting the knowledge in the *Linked Data*. Next, (Section 3.2) given $W_{j,i}$ the set of Web pages from a particular website j containing entities of the concept c_i, each $d_{i,k}$ is used to annotate $W_{j,i}$ by matching every entry in $d_{i,k}$ against the text content in the leaf nodes on a web page.

This process produces a set of annotations on each web page. We define an annotation as a pair $< xpath, value_{i,k} >$, where $xpath$ is the DOM tree xpath to the labelled node on the web page, and $value_{i,k}$ is the text content of the node, which matched an entry in the dictionary $d_{i,k}$. Due to the structural consistency of Web pages on a website, we expect to see the same or similar set of xpaths across all Web pages.

On the other hand, due to the diversity in attribute values, we expect a good candidate xpath to extract different values for each attribute across a website (high level of variance). Therefore, in the final step (Section 3.3), for each attribute, the different xpaths are gathered and their matched values in the corresponding attribute dictionary are counted. The confidence of each xpath is rated based on the number of different values it extracts, and the best candidate xpaths are selected to be the wrapper for that attribute on the specific website, denoted as $wp_{j,i,k}$. The wrapper is then applied to re-annotate the website j for attribute $a_{i,k}$.

3.1 Dictionary generation

As mentioned before, the *Linked Data* contains billions of facts for specific domains, and can be used as a large entity knowledge base in many tasks [10, 14]. In the first step, for each attribute of each concept, we exploit *Linked Data* to create a dictionary for that attribute and use it to annotate Web pages. The goal is to create a sufficiently large gazetteer that is a good representation of the task of interest.

The challenge at this step is to translate the concept (e.g., university) and attributes (e.g., name) of interest to the vocabularies used within the *Linked Data*, such that the corresponding knowledge can be identified. We define this process as "user information need formalisation". This process is non-trivial because on one hand, it is known that the vocabularies used on the *Linked Data* are highly heterogeneous [17]. For example, both `http://dbpedia.org/ontology/University` and `http://schema.org/CollegeOrUniversity` can be used to represent the concept of "university", however, they are not necessarily mapped to each other by data publishers. As a result, finding information within *Linked Data* by structured queries requires users to not only be familiar with the underlying *Linked Data* vocabularies, but also the data structure and query language in order to be able to translate the terms and retrieve data. On the other hand, natural language is highly ambiguous. Description of a concept at phrase-level is sometimes insufficient to identify the correct concept. For this reason, many natural language based approaches that automatically translate such information needs into structured queries [13] can fail and incorrect data may be retrieved.

While this brings a new series of research questions, the solution in this study is based on a simplified scenario. We assume that users are familiar with the SPARQL language, a universal structured query language to access *Linked Data* on the Web. And we require users to manually explore the vocabularies used in *Linked Data* and retrieve relevant data using SPARQL queries. Whenever possible, we limit our choices to two mainstream vocabularies: `dbpedia.org` and `schema.org`. In future work, we shall research methods to automate this process.

Thus under the simplified scenario, our goal can be achieved in three steps. First, given a SPARQL endpoint, we query the exposed *Linked Data* to identify the relevant concepts; second, we manually select the most appropriate class and properties that describe the attributes of interest; third, using the SPARQL endpoint we query the *Linked Data* to retrieve instances of the properties of interest. For example, the following query[3] can find all concepts matching the keyword "university"within the data:

```
SELECT DISTINCT ?uni
WHERE {
?uni rdf:type owl:Class ;
rdfs:label ?lab .
FILTER regex(?lab,"university","i") }
```

Table 1: Example concepts to describe "university"

http://dbpedia.org/ontology/University
http://data-gov.tw.rpi.edu/vocab/c/University
http://semanticweb.org/id/Category-3AUniversity

We manually check the results and select `http://dbpedia.org/ontology/University` as the best candidate. Next, we query *Linked Data* to identify all properties defined with this concept using the query below, and manually select `http://dbpedia.org/property/name` to match the attribute 'university name'.

```
SELECT DISTINCT ?prop
WHERE {
?uni a <http://dbpedia.org/ontology/University> ;
?prop ?o . }
}
```

Finally, we write a SPARQL query to extract all available values of this attribute. Following this example, this would be:

```
SELECT DISTINCT ?name
WHERE{
?uni a <http://dbpedia.org/ontology/University> ;
<http://dbpedia.org/property/name> ?name .
  FILTER (langMatches(lang(?name), 'EN')).
```

The query returns a list of distinct values of `http://dbpedia.org/property/name` of all instances of `http://dbpedia.org/ontology/University`. These are used as a dictionary for university names in the following annotation process. Note that the dictionary generation process is completely independent from the data. No a priori knowledge about the data is introduced to this process and thus the dictionary is unbiased and universal for any extraction tasks concerning the attribute "university name".

[3]Other queries can also achieve the same objective. This is just an example.

3.2 Web page annotation

The dictionaries thus created are used to annotate each website in this step. To do so, each Web page is parsed into a DOM tree and the leaf node that contains texts along with its XPath are identified. Then to annotate for a particular attribute $a_{i,k}$, each candidate in $d_{i,k}$ is matched against the text content of each node. If there is an exact match between the text content and any candidate in the dictionary, the XPath and the corresponding node are labelled by the attribute and a pair $< xpath, value_{i,k} >$ is created for this website. This annotation process is completely unsupervised and generic, since it does not require any a priori knowledge about the data or task.

3.3 XPath identification and re-annotation

The annotation step produces a set of $< xpath, value_{i,k} >$ pairs for each website. There are two problems with these annotations. First, due to the incompleteness of the auto-generated dictionaries, the annotation process may not cover the entire data set and the number of false negatives can be large (i.e., low recall). However, we expect the large dictionaries to cover more than enough to learn useful wrappers for the attributes. Second, entries in the dictionaries can be ambiguous (e.g., 'Home' is a book title that matches part of navigation paths on many Web pages) but annotation does not involve disambiguation.

Therefore in this step, we analyse the annotations and identify best candidate *xpaths* as wrappers which will be used to re-annotate the website. For a particular attribute $a_{i,n}$ and a website collection, we gather all annotations ($< xpath, value_{i,k} >$), find the distinct *xpath* of them, and create a mapping between *xpath* and the set of distinct values matched by that *xpath* across the entire website collection. Thus an entry in the map is a pair $< xpath, value_{i,k} | k = n >$ where n denotes the attribute of interest is $a_{i,n}$. Based on the hypothesis of structural consistency in a website, we expect the majority of true positives to share the same or similar *xpath*. Also, since an attribute is likely to have various distinct values, the top ranked $< xpath, value_{i,k} | k = n >$ pairs by the size of $value_{i,k} | k = n$ are likely to be useful XPaths for extracting the attribute $a_{i,n}$ on this website collection.

In this work, we simply select the highest ranked *xpath* to be the wrapper for the attribute on the specific website. Although research has shown that often Web pages from the same website can have slight structural variations, causing a single wrapper to fail at times, enhancing wrapper robustness is another challenging research question that often requires complicated modelling and computation [5, 6, 8]. In this work, we do not aim to address this issue, but show that for large scale extraction tasks, this simple approach can achieve very competitive results on certain data.

Consequently the Web page annotation and XPath identification are repeated for every attribute on every website collection, creating a wrapper for each attribute-website pair. Finally, each wrapper is applied to re-annotate the website for the corresponding attributes.

4. DATASET AND EVALUATION METHOD

The dataset we used for the experiments has been constructed and made publicly available by Hao et al. [8]. It consists of around 124K pages collected from 80 websites. These websites are related to 8 verticals, including *Autos, Books, Cameras, Jobs, Movies, NBA Players, Restaurants, and Universities.* For each vertical they collected detail pages from 10 different websites (200 to 2,000 pages per website). For each vertical they selected a set of 3 to 5 common attributes to extract. Table 2 shows the statistics of the dataset, where *WS* shows the number of different websites, *WP* shows the total number of web pages.

Table 2: Gold Standard dataset statistics

Vertical	WS	WP	Attributes
Auto	10	17923	model (m), price (p), engine (e), fuel economy (f)
Book	10	20000	title (t), author (a), ISBN-13 (i), publisher (p), publish-date (pd)
Camera	10	5258	model (md), price (p), manufacturer (m)
Job	10	20000	title (t), company (c), location (l), date (d)
Movie	10	20000	title (t), director (d), genre (g), rating (r)
NBA player	10	4405	name (n), team (t), height (h), weight (w)
Restaurant	10	20000	name (n), address (a), phone (p), cuisine (c)
University	10	16705	name (n), phone (p), website (w), type (t)

A ground truth has been created by Hao et al.[8] for this dataset. For each attribute-website pair, a file listing all possible attribute values found on the website is generated. The values are discovered by using a few handcrafted regular expressions over each website. Note that for some verticals (e.g., university), certain websites do not list all attributes of the vertical (e.g., phone number is not listed on *collegetoolkit.com.* There are in total 5 such cases in the dataset.

Each attribute can have none or one to multiple correct answers. In some cases, an attribute has mulitple answers because of the different lexicalisation of the single value on the web page (e.g., Volkswagen Group, Volkswagen Group Plc.). In other cases, an attribute is a multi-valued attribute and can have several possible answers (e.g., book authors). In [8], for multiple answers, a prediction is counted correct as long as at least one of the answers in the groundtruth is extracted. In our experiment, we use the same strategy. The same evaluation metric (F1) is also used.

5. EXPERIMENT AND RESULTS

In this section, we describe experiments designed to evaluate the proposed method. As described before, our method depends on the availability of a suitable dictionary for an attribute of interest. Our first goal is to evaluate the performance of the method under ideal conditions. To do so, we create dictionaries specifically tailored to the data. These dictionaries are expected to have the minimum level of noise, and therefore, sets a higher limit of the performance of the method. We call this experiment *topline.* Next, we follow the method described in Section 3.1 to generate dictionaries exploiting *Linked Data*, and re-apply the method to annotate the data. We refer to this experiment as *Linked Data (ld) based WI* and we use this result to compare against the topline and state of the art. Since the dictionaries in the second experiment are generated independently from the data, they are likely to contain noise. Thus the second set of experiments not only further evaluates the method of Wrapper Induction, but also provides an indication of the robustness of the method with respect to noise.

5.1 Generating dictionaries

To generate the topline dictionaries, for each attribute of a vertical we simply collect all answers in the ground truth to build a dictionary for that attribute. This makes sure that when annotating each website all (but not only) the true answers are contained in the dictionary. To generate *Linked Data* based dictionaries, we follow the methodology described in Section 3. That is, we manually explore *Linked Data* and create queries to retrieve instances to populate attribute dictionaries. We used Sindice SPARQL endpoint[4] to retrieve information from the *Linking Open Data* cloud. Table 3 shows the size of each dictionary obtained by the two different methods. For some verticals and attributes we were not able to generate a dictionary from the *Linked Data* as we were not able to find a suitable concept on the *Linking Open Data* cloud to represent the vertical. When comparing the result we will only consider the attributes for which we did generate a *Linked Data* based dictionary.

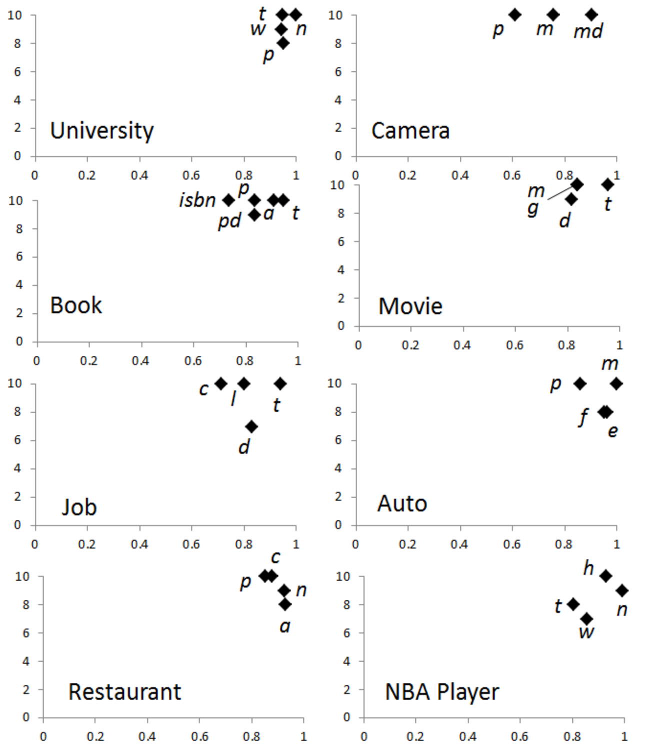
Table 3: Attribute dictionary statistics

Vertical	Attribute	topline	linked data
University	phone	16973	283
	website	7968	12930
	name	9224	13144
	type	68	
Camera	model	5428	
	price	1524	
	manufacturer	253	
Book	isbn_13	19302	39112
	author	14228	13060
	title	17402	37485
	publication date	6645	3048
	publisher	6175	520
Movie	genre	1398	114
	title	17146	57292
	mpaa rating	3255	2
	director	7398	16079
Job	title	17712	
	date posted	2381	
	location	5634	
	company	5655	
Auto	model	9916	
	price	10792	
	engine	2469	
	fuel economy	2051	
Restaurant	phone	19510	
	cuisine	2378	72
	address	29687	37
	name	16631	312
NBA player	weight	507	
	height	121	
	name	1457	9194
	team	60	677

5.2 Results

The two different sets of attribute dictionaries are then used to induce wrappers for each attribute-website pair following the method described in Section 3. The induced wrappers are then applied to re-annotate the website collection for the attributes to create final annotations. We noticed that, while in a majority of cases the induced wrappers achieved very high accuracy, there are also a number of cases where they failed (i.e., an incorrect wrapper is induced and no or very few true positives are annotated). Further analysis has shown that such failures are often related to the nature of specific websites, which we generalise as situations in which the proposed method are unsuitable. This will be further discussed in the later sections. In presenting the results, we only consider situations when true positives (at least one) are extracted by the learnt wrappers. Thus in Figure 1, we show the average accuracy for each vertical-attribute across all websites for the *topline*. The x-axis shows the accuracy (F1) and the y-axis shows the number of websites for which an attribute can be extracted by the induced wrapper. For brevity, we call this "coverage". The lower case letters are shorthands to represent the attributes of verticals (see Table 2 for keys). Take *book* for example, Figure 1 shows that the wrappers for the *title (t)* attribute induced by our method can extract true positives from all of the 10 websites, with an average accuracy of approximately 98%. In the following, we will say that our Wrapper Induction method can *cover* 10 websites for this attribute. As another example, for *publication date (pd)*, the induced wrappers extracts true positives on 9 websites (or covers 9 websites), with an average accuracy of 83%. Similarly, Figure 2 shows the same kind of information but for the experiments using the *Linked Data* generated dictionaries.

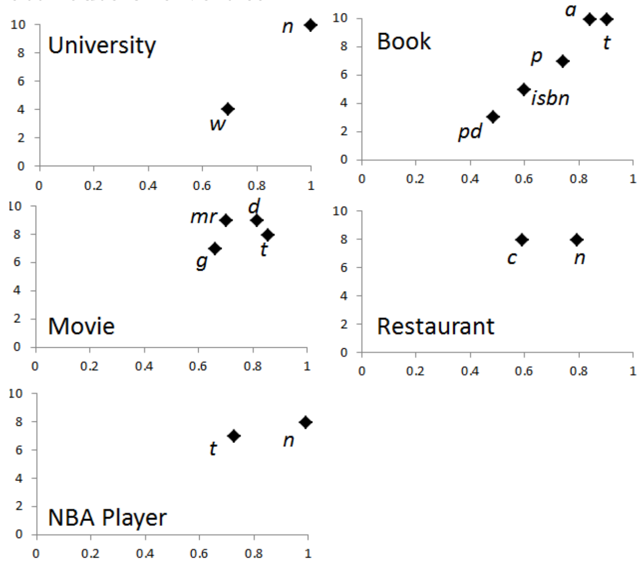
Figure 1: The F1 accuracy of *topline* for each attribute of a vertical.



In Figure 3, we show the average accuracy for each vertical-website combination, summing up all attributes of the vertical. The number in brackets shows the number of websites covered for the vertical and each point on the graph represents a website. The x-axis shows accuracy (F1) and the y-axis shows the number of attributes of the vertical. As an example, Figure 3 shows that for *book*, all of the 10 websites can be covered by our method and for 9 of them, all the 5 attributes can be extracted. The average accuracy for all attributes ranges from 50% to 100%, with most websites reaching very high accuracy of 90∼100% (as many points clutter at the top right corner in the graph). Similarly, we

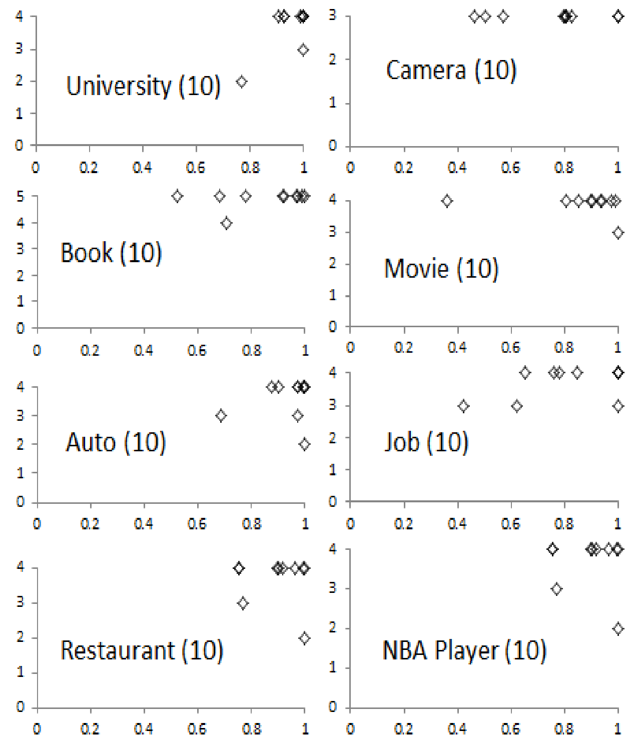
[4]`http://sparql.sindice.com/`

Figure 2: The F1 accuracy of *ld-based WI* for each attribute of a vertical.



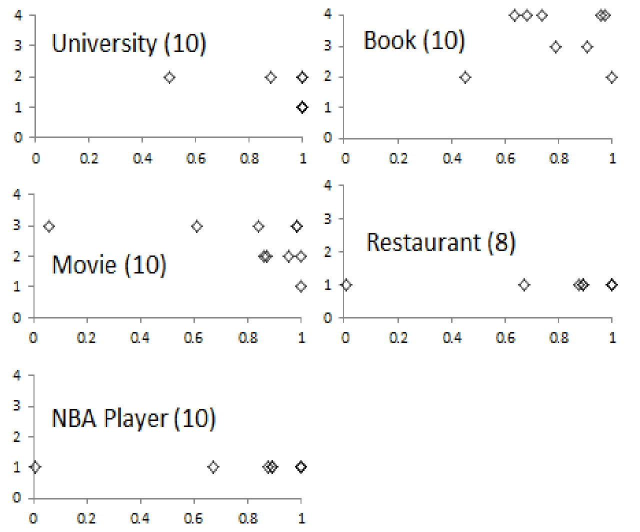
show the same kind of information for the experiments using the *Linked Data* generated dictionaries in Figure 4.

Figure 3: The F1 accuracy of *topline* for each website of a vertical



Finally, in Table 4, we compare both the results of *topline* and the *ld-based WI* against [8]. It is known that [20] also used the same dataset for performing experiments on Wrapper Induction. We do not report their figures in Table 4 because they are not directly comparable, as the authors use

Figure 4: The F1 accuracy of the *ld-based WI* for each website of a vertical.



a only a derived version of the dataset, selecting samples of pages from 4 of the 8 verticals.

Table 4: Comparison of F1 per vertical

Concept	Hao[8]	*topline*	*lod*
auto	0.71	0.94	
book	0.87	0.85	0.78
camera	0.91	0.76	
job	0.85	0.82	
movie	0.79	0.86	0.76
nbaplayer	0.82	0.9	0.87
restaurant	0.96	0.89	0.69
university	0.83	0.96	0.91

6. DISCUSSION

6.1 Interpretation of results

First, Figure 1 and Figure 3 give an indication of whether our simple dictionary-based wrapper induction approach is effective, under the condition that perfect dictionaries can be obtained for the task of interest. As it is shown in Figure 1, this simple approach is quite effective in most cases where an accuracy of >80% can be achieved; indeed there are a number of cases where it achieved 100% accuracy.

Figure 3 shows that, with perfect dictionaries, the performance of the method varies across different websites. For example, for university, there is one website on which only two attributes can be extracted; for book, there are three websites on which the accuracy is below 80% and varies significantly.

Next, Figure 2 and Figure 4 show the results obtained with dictionaries automatically generated using *Linked Data*. Despite the gigantic size of the Linking Open Data (LOD) cloud, we show that in this particular experiment, there are still domains that are not well represented as *linked open data*. For example, we were not able to create dictionaries for Camera and Auto, because 1) we could not find any authoritative vocabularies within the *LOD* cloud to describe them, and 2) the only available vocabularies do not have instances defined on the *Linked Data*. Furthermore, existing knowledge on the LOD cloud can be highly noisy. For example, the drop in accuracy in the attribute *genre* for movie is largely because of the incorrect candidates retrieved from the *Linked Data*. The query for fetching instances of `http://dbpedia.org/property/genre` that is a property of `http://dbpedia.org/ontology/film` returned 114 instances where above 95% are "music" genres such as "pop" and "rock".

Additionally, by comparing the results against the *topline*, we found that the quality of dictionaries indeed has an impact on the wrapper induction process. Also the attributes that suffer most seem to be those that are highly variant, can contain infinite possible values, such as telephone numbers, addresses, prices and dates. Note that some of these attributes are not shown on the figures since the wrapper induction process failed to generate useful wrappers that can extract true positives.

Nevertheless, by comparing figures in Table 4, we note that our method can still be very competitive on data that it is suitable for. We believe this is encouraging considering the very simple and unsupervised nature of our approach. Thus an important question is to answer under what situations the method is most effective, which we discuss below.

6.2 Lessons learnt

Based on the above observations we conclude that the proposed method can be highly successful under certain conditions. The main condition for success is the regularity in terms of structure of the website. In the considered dataset this assumption is true on the majority of cases, obtaining an overall accuracy of 87%. On web sites with a more variable nature, i.e. where the same information is positioned in different places for different pages, the method is not robust. This is a direct consequence of the naive way of choosing the winning xpath, as the strategy we present simply picks the first ranking xpath in term of dictionary coverage. The Amazon website for example is one for which the method has poor performance both on the Book domain and the Camera domain. In a book page e.g. the target title can be found in slightly different positions (`/HTML[1]/BODY[1]/DIV[7]/H1[1]/SPAN[1]/text()[1]`, `/HTML[1]/BODY[1]/DIV[8]/H1[1]/SPAN[1]/text()[1]`...) and other book titles other than the one of the target entity are located in several areas of the page (`/HTML[1]/BODY[1]/DIV[20]/DIV[1]/FORM[1]/FIELDSET[1]`, `/HTML[1]/BODY[1]/A[6]/text()[1]`,...). For these cases finding a way to pick all the relevant xpaths would improve the performance.

To prove this idea we tested additional strategies to chose which xpath (or xpaths) is (are) the best for each website to extract the information we are looking for. We tried (i) partial matching of the xpath (i.e. relaxing position fillers in the path), (ii) taking multiple xpath (up to covering a certain percentage of annotations) and (iii) using multiple xpaths, but only if similar to the single winning one, apart from position fillers. The latter, when applied to extraction of book title from the Amazon website, produced an accuracy of 95% (against 45% when applying the simple strategy). We did not investigate this issue further as the focus of the work is not finding ways of improving knowledge based Wrapper Induction per se, but proving that in the cases where it is likely to work, we can use *Linked Data* generated resources. Therefore we went forward on the construction of *Linked Data* based dictionaries. An obstacle we encountered in this process was the difficulty of retrieving concepts of interest, when not familiar with the vocabularies on the *Linked Data*. For some of the concepts we could not find a specific appropriate concept in the *Linked Data* to express the information need. These issues will be addressed in future work with dedicated experiments on formulation of SPARQL queries.

Once obtained the dictionaries (in Table 3), we reapplied the method using those. For covered concept-attributes the experiments has an accuracy comparable with the *topline* experiment. The behaviour of the method using *Linked Data* based dictionaries is consistent with the *topline* dictionaries: poor performances appear to be related to the website structure rather than the quality of the dictionary, as the decrease of performances happens mostly for the same websites in both experiments. The method is indeed robust to poor quality of the dictionary. As an example, the dictionary which we generate for the attribute *genre* of the concept *movie* is rather semantically dubious. If querying for the property `http://dbpedia.org/property/genre` of the concept `http://schema.org/Movie`, returned values mostly refer to the genre of the movie soundtrack than the genre of the movie, which is a result of the triple extraction process from Wikipedia to DBpedia [10]. We manually checked the dictionary entries (114) for *genre*. 111 of them refers to music genre (e.g. Indie pop, Country music, Drum and bass, Groove metal, Indie rock ...), while only 3 of them refer to actual movie genre (Philosophical fiction, Thriller, Western). Nevertheless, the performance of the method for this attribute are perfectly comparable with *topline*. The failure (average accuracy per attribute of 6%) for *allmovie* website also happens with the *topline* experiment (average accuracy per attribute of 36%), so it is likely to be related to the website structure.

With respect with the state of the art the overall accuracy of the the method proposed by [8] is 84% (calculated by macro-averaging the figures reported on the paper), while our method obtains an accuracy of 80% on covered concepts and attributes. The result is encouraging considering that our method is (i) very simple, do not require any kind of pre-processing of the web pages or collection of additional features (ii) fully unsupervised.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a knowledge based method for *Wrapper Induction*. The simple idea behind the method is to (i) generate knowledge resources, in the form of dictionaries, for each type of information to extract and (ii) use the dictionaries to annotate websites and discover if any recurrent pattern exists to locate the information on the website.

The major contribution of this work is a study on the suitability of *Linked Data* to build each specific dictionary for Wrapper Induction, and experiments on a publicly available dataset show encouraging results for covered cases, with an average accuracy of over 80%, although some failure cases occur. The main advantage of the method is that it does not require any training material for Wrapper Induction. A dictionary is built once for each type of information to extract and it is reusable across all websites of a pertinent domain.

The implication is that adapting the method across domains and websites will require little human effort.

One limitation of the approach in the dictionary generation phase is represented by cases where the concept and knowledge instances are not present or not easy to locate in the *Linked Data*. To address this, we will look into the research of information retrieval on the semantic web to identify possible solutions. Also, we aim to study methods that can assist human users to define their information needs and retrieve relevant knowledge in *Linked Data* in a semi-automatic way.

Another limitation is the lack of robustness in the learnt wrappers. Experiments have shown that two major failure situations are due to (i) the irregular structure of the website or (ii) the quality of the dictionary. To address (i), future work will focus on defining a strategy to decide when to use a *single xpath*, *multiple xpaths*, or *partial matching of the xpath* depending on certain prior knowledge about the vertical and attribute (e.g., whether the attribute is single-valued or multi-valued). Furthermore, we aim to find solutions that can derive such knowledge in a (semi)automatic way. The problem regarding the quality dictionaries will require more investigation. As a first step, we aim to propose measures to assess whether a dictionary is sufficiently large for a task.

Acknowledgments

Part of this research has been sponsored by the EPSRC funded project LODIE: Linked Open Data for Information Extraction, EP/J019488/1

8. REFERENCES

[1] A. Arasu and H. Garcia-Molina. Extracting structured data from web pages. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 337–348. ACM, 2003.

[2] R. Blanco, H. Halpin, D. Herzig, and P. Mika. Entity search evaluation over structured web data. In *SIGIR 2011*, 2011.

[3] A. Carlson and C. Schafer. Bootstrapping information extraction from semi-structured web pages. *e European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2008.

[4] V. Crescenzi and G. Mecca. Automatic information extraction from large websites. *Journal of the ACM*, 51(5):731–779, Sept. 2004.

[5] N. Dalvi, P. Bohannon, and F. Sha. Robust web extraction: an approach based on a probabilistic tree-edit model. *Proceedings of the 35th SIGMOD international conference on Management of data*, 2009.

[6] N. Dalvi, R. Kumar, and M. Soliman. Automatic wrappers for large scale web extraction. *Proceedings of the VLDB Endowment*, 4(4):219–230, 2011.

[7] P. Gulhane, A. Madaan, R. Mehta, J. Ramamirtham, R. Rastogi, S. Satpal, S. H. Sengamedu, A. Tengli, and C. Tiwari. Web-scale information extraction with vertex. *2011 IEEE 27th International Conference on Data Engineering*, pages 1209–1220, Apr. 2011.

[8] Q. Hao, R. Cai, Y. Pang, and L. Zhang. From One Tree to a Forest : a Unified Solution for Structured Web Data Extraction. In *SIGIR 2011*, pages 775–784, 2011.

[9] T. Heath and C. Bizer. Linked data: Evolving the web into a global data space. *Synthesis Lectures on the Semantic Web: Theory and Technology*, 1(1):1–136, 2011.

[10] G. Kobilarov, C. Bizer, S. Auer, and J. Lehmann. DBpedia-A Linked Data Hub and Data Source for Web and Enterprise Applications. In *WWW2009*, pages 1–3, 2009.

[11] S. Krause, H. Li, H. Uszkoreit, and F. Xu. Large-scale learning of relation-extraction rules with distant supervision from the web. In *Proceedings of the 11th international conference on The Semantic Web - Volume Part I*, ISWC'12, pages 263–278, Berlin, Heidelberg, 2012. Springer-Verlag.

[12] N. Kushmerick. Wrapper Induction for information Extraction. In *IJCAI97*, pages 729–735, 1997.

[13] V. Lopez, M. Fernández, E. Motta, and N. Stieler. Poweraqua: Supporting users in querying and exploring the semantic web. *Semantic Web*, 3(3):249–265, 2012.

[14] V. Mulwad, T. Finin, Z. Syed, and A. Joshi. Using linked data to interpret tables. In O. Hartig, A. Harth, and J. Sequeda, editors, *COLD*, volume 665 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010.

[15] I. Muslea, S. Minton, and C. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*, pages 1–28, 2001.

[16] I. Muslea, S. Minton, and C. Knoblock. Active Learning with Strong and Weak Views : A Case Study on Wrapper Induction. *IJCAI'03 8th international joint conference on Artificial intelligence*, pages 415–420, 2003.

[17] A. Nikolov, V. Uren, E. Motta, and A. Roeck. Overcoming schema heterogeneity between linked semantic repositories to improve coreference resolution. In *Proceedings of the 4th Asian Conference on The Semantic Web*, ASWC '09, pages 332–346, Berlin, Heidelberg, 2009. Springer-Verlag.

[18] A. Parameswaran, N. Dalvi, H. Garcia-Molina, and R. Rastogi. Optimal Schemes for Robust Web Extraction. In *37th International Conference onVery Large Data Bases*, 2011.

[19] S. Soderland. Learning information extraction rules for semi-structured and free text. *Mach. Learn.*, 34(1-3):233–272, Feb. 1999.

[20] D. Song, Y. Wu, L. Liao, L. Li, and F. Sun. A dynamic learning framework to thoroughly extract structured data from web pages without human efforts. *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics - MDS '12*, l:1–8, 2012.

[21] C. Welty, J. Fan, D. Gondek, and A. Schlaikjer. Large scale relation detection. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, FAM-LbR '10, pages 24–33, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[22] T. Wong and W. Lam. Learning to adapt web information extraction knowledge and discovering new attributes via a Bayesian approach. *Knowledge and Data Engineering, IEEE*, 22(4):523–536, 2010.