FOCIH: Form-Based Ontology Creation and Information Harvesting — Source link ↗

Cui Tao, David W. Embley, Stephen W. Liddle

Institutions: Brigham Young University

Related papers:

- Performance motivation systems and methods for contact centers

- Methods and systems for object-based relational distributed databases

- Personal document container

- Method and apparatus for ontology matching

- Electronic Documents Signing and Compliance Monitoring Invention

FOCIH: Form-based Ontology Creation and Information Harvesting

Cui Tao[*][†], David W. Embley[*][†], and Stephen W. Liddle[‡]

[†]Department of Computer Science
[‡]Information Systems Department
Brigham Young University, Provo, Utah 84602, U.S.A.

Abstract. Creating an ontology and populating it with data are both labor-intensive tasks requiring a high degree of expertise. Thus, scaling ontology creation and population to the size of the web in an effort to create a web of data—which some see as Web 3.0—is prohibitive. Can we find ways to streamline these tasks and lower the barrier enough to enable Web 3.0? Toward this end we offer a form-based approach to ontology creation that provides a way to create Web 3.0 ontologies without the need for specialized training. And we offer a way to semi-automatically harvest data from the current web of pages for a Web 3.0 ontology. In addition to harvesting information with respect to an ontology, the approach also annotates web pages and links facts in web pages to ontological concepts, resulting in a web of data superimposed over the web of pages. Experience with our prototype system shows that mappings between conceptual-model-based ontologies and forms are sufficient for creating the kind of ontologies needed for Web 3.0, and experiments with our prototype system show that automatic harvesting, automatic annotation, and automatic superimposition of a web of data over a web of pages work well.
Keywords: ontology generation from forms, information harvesting from the web, automatic annotation of web pages, web of data, Web 3.0.

1   Introduction

Many see the next generation web (Web 3.0) as a web of data in which users query for facts directly rather than use search engines to find pages that contain facts. A major impediment to this Web 3.0 vision is content creation. Creating the required ontologies and populating them with data yields a web of data, but both ontology creation and ontology population are human-intensive tasks requiring a high degree of expertise.

To alleviate this problem, researchers are developing ways to make Web 3.0 creation "human scalable." Typifying this desire, the *Journal of Web Semantics* recently called for papers on "human-scalable and user-friendly tools that open the Web of Data to the current Web user." Efforts to create user-friendly, web-scalable tools are on the agenda of many research labs around the world.

Researchers are interested both in easing the burden of ontology creation and in automatic semantic annotation:

- With regard to easing the burden of manual ontology creation (e.g., via Protege [23] or OntoWeb [28]), researchers are developing semi-automatic ontology generation tools. Tools such as OntoLT [6], Text2Onto [9], OntoLearn [22], and KASO [35] use machine learning methods to generate an ontology from natural-language text. These tools usually require a large training corpus, and, so far, the results are not very satisfactory [24]. Tools such as OntoBuilder [12], TANGO [32], and the ones developed by Pivk et al. [24] and Benslimane et al. [5] use structured information (HTML tables and forms) as a source for learning ontologies. Structured information makes it easier to interpret new items and relations. These approaches, however, derive concepts and relationships among concepts from source data, not from users, and thus do not provide the control some users need to express the ontological world-views they desire.
- With regard to enabling automatic annotation, typical approaches (e.g., [2, 4, 8, 10, 15, 17, 21, 33]) base their work on information extraction [26]. Post-extraction alignment with ontologies, however, is their main drawback [17]. A way to overcome this drawback is through "extraction ontologies"—ontologies with data recognizers that are able to directly and automatically extract and thus annotate data with respect to specified ontologies (e.g., [11, 18, 19]). Extraction ontologies, however, rely on human expertise to manually create, assemble, and tune reference sets and data recognizers. In another direction that tends to overcome both the alignment drawback and the manual-creation drawback, researchers propose structuring unstructured data for query purposes [7] or doing "best-effort" information extraction [27]. These approaches, however, yield less precise results both for the ontological structure of the data and for the annotation of the data with respect to the ontological structure.

We created FOCIH (Form-based Ontology Creation and Information Harvesting, pronounced *foh*·sī) to accomplish three goals: (1) to ease the burden of manual ontology creation while still giving users control over ontological views; (2) to enable automatic annotation that aligns with user-specified ontologies, does not require manual creation of extraction ontologies, and is precise; and (3) to facilitate the semi-automatic construction of a web of data. The form-based part of the FOCIH name emphasizes the means by which a user creates an ontology—namely by creating a form. The information-harvesting part of the name emphasizes FOCIH's ability to harvest information by automatically filling in the form for each page in a web site containing machine-generated display pages (usually hidden-web-site pages). FOCIH provides for semi-automatically annotating information according to any view users want—thus opening a pathway to the envisioned Web 3.0.

We present the details of these contributions as follows. Section 2 describes how users create forms and annotate a sample page by filling in the form. Section 3 explains how FOCIH generates ontologies based on user-created forms.

Fig. 1. An Filled in Form with a Source Data Page

Section 4 discusses path and instance recognition which allows FOCIH to automatically harvest and annotate information with respect to the created form and thus semantically annotate web pages with respect to the ontology generated from the form. Section 5 describes our experiences with our prototype, including experimental performance measurements. Section 6 presents current and future work: reverse-engineering tables and ontologies for FOCIH form initialization; initial population of FOCIH forms using information-extraction ontologies; and synergistic creation of information-extraction ontologies—all in an effort to further reduce, and in some cases entirely eliminate, the work required of the knowledge worker. In Section 7, we make concluding remarks about FOCIH and tie in the work presented here with a vision of how to create Web 3.0—a vision of how to automatically superimposition of a web of data over a web of pages.

2   Form Creation and Annotation

The FOCIH GUI has two modes of operation: form creation and form annotation. Form creation allows users to create forms in accord with how they wish to organize their information. Form annotation allows users to annotate pages with respect to created forms. We use the form about countries and the web page for the Czech Republic in Figure 1 as a running example. The screen shot in Figure 1 shows the running example at the end of the form-annotation mode.

FOCIH has five basic form elements from which users can choose: *single-label/single-value element* (⊶⊡), *single-label/multiple-value element* (⊟), *multiple-label/multiple-value element* (⊞), *mutually-exclusive choice element* (⅜⋮),

and *non-exclusive choice element* (.). A user begins with an empty base form with only a place for a form title and icons (⬚ ▦ ▦ ▦ ▦) for each of the form elements. The user can edit the title; Figure 1 shows that the user has chosen *Country* as the title for the form. Clicking on a form-element icon causes the element to appear. The user then has control to edit form labels.

Thus, for the single-label/single-value elements in Figure 1, the user has clicked on the single-label/single-value icon6 (⬚) and has then labeled the form element *Name*, and has again clicked on the icon and labeled the form element *Capital*, and again for *Geographic Coordinate*. (At this point in form creation, the form fields in the elements would still be empty as is *Geographic Coordinate* field in Figure 1.)

For the single-label/multiple-value element in Figure 1, the user has clicked on the single-label/multiple-value icon (▤) and has labeled the element *Religion*. Multiple-label/multiple-value elements are similar except that they have multiple columns. The user can expand/contract the number of columns as desired. In Figure 1, the user has created a multiple-label/multiple-value element with two columns, *Population* and *Year*.

Choice elements let users specify decompositions of concepts. In Figure 1, the user has created a non-exclusive choice element labeled *Life Expectancy*. The decomposition of life expectancy is by *Male Life Expectancy* and *Female Life Expectancy*. The decomposition form fields are *non-exclusive* because the set of values over all countries for male life expectancy can have a non-empty intersection with the set of values for female life expectancy.

Area in Figure 1 illustrates the possibility of nesting form elements inside one another. *Area* is a single-label/single-value form element. Nested within it, the user has nested a form with three single-label/single-value form elements: *Water*, *Land*, and *Total*. In general, users can nest an entire form inside any other form element. The nesting can continue to any depth.

Users annotate a page from a web site with respect to a created form by filling in the form. For example, to annotate the string "Prague" as the *Capital* of the Czech Republic, a user drags the mouse cursor over "Prague" to highlight it in the source and then clicks on the pencil icon (✎) in the single-entry *Capital* field. FOCIH adds 'Prague" to the form field under *Capital* as Figure 1 shows.

The user can add multiple values in a multiple-value element by highlighting and adding each, one by one. The user must be careful to put related values in the same row for multiple-column form elements. For example, the user must put *Population* 10,264,212 and *Year* 2001 in the same row as Figure 1 shows.

The user can also concatenate two or more highlighted values to form a single value in the form. After placing the first value in a form field, the user highlights the second (third, ...) and clicks on the plus icon (⊞) rather than the pencil icon. For example, suppose a web site displays *Geographic Coordinate* information by listing longitude and latitude separately, but the user wants them combined into a single compound value. The user would first enter the longitude value in the *Geographic Coordinate* field in the usual way and then highlight the latitude
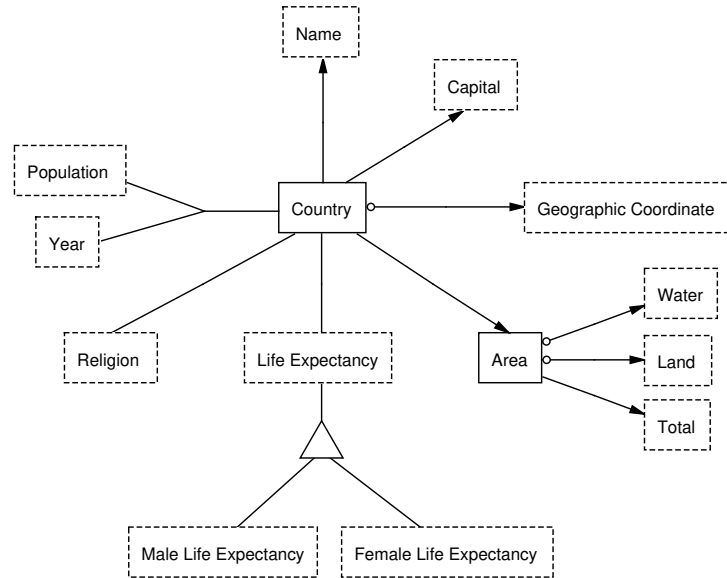
Fig. 2. The Graphical View of a Sample Ontology

value and click on the plus icon (rather than on the pencil icon) in the *Geographic Coordinate* field.

3   Ontology Generation

From a created form, FOCIH can infer and generate an ontology. Figure 2 shows a generated ontology for the form in Figure 1. We use OSM [11] as the conceptual-model basis for an extraction ontology. The advantage of OSM is that it has a high-level graphical representation that directly translates to predicate calculus. Thus, when appropriately limited, it translates in a straightforward way to OWL [34] and to various description logics [3]. Even more important, however, is OSM's ability to support data extraction from source documents [11].

Based on the from title, FOCIH generates a non-lexical[1] concept with this title as the name. Thus, for the form in Figure 1, FOCIH generates the concept *Country* as Figure 2 shows. Every label in the form also represents a concept in the corresponding ontology; the label is the name for the concept. Form concepts with nested components become non-lexical object sets. Thus, *Area* is non-lexical. Form concepts without nested components become lexical object sets. Thus, the remaining concepts are all lexical. As a consistency requirement, generalization/specialization concepts must all be lexical or must all be non-lexical.

---

[1] Lexical concepts, represented by dashed boxes, are for values that represent themselves; non-lexical concepts, represented by solid boxes, are for object identifiers that represent real-world objects such as countries in our running example.

To meet this requirement, FOCIH declares all the object sets involved in a generalization/specialization to be lexical if there are no nested components other than the nesting of generalization/specialization components themselves; otherwise all concepts are non-lexical. Thus, *Life Expectancy*, *Male Life Expectancy*, and *Female Life Expectancy* are all lexical.

FOCIH generates relationship sets[2] among the concepts as follows.

Single-label/single-value form elements. Between the form-title concept *T* and each top-level single-label/single-value form element *S*, FOCIH generates a functional binary relationship set from *T* to *S*. Thus, FOCIH generates functional relationship sets from *Country* to *Name*, *Capital*, *Geographic Coordinate*, and *Area* respectively as Figure 2 shows. Similarly, between each form element *E* and a single-label/single-value form element *S* nested inside *E*, FOCIH also generates a functional binary relationship set from *E* to *S*. Thus, FOCIH generates functional relationships from *Area* to *Water*, *Land*, and *Total* respectively.

Single-label/multiple-value form elements. Between each form-title concept *T* and each single-label/multiple-value concept *M*, FOCIH generates a non-functional binary relationship set between *T* and *M*. Thus FOCIH accommodates the possibly many *Religions* for each *Country* as Figure 2 shows. Although our running example has no nested single-value/multiple-value form elements, FOCIH also creates non-functional binary relationship sets between a parent form element and each nested child single-label/multiple-value form element.

Multiple-label/multiple-value form elements. Between the form-title concept and each multiple-label form element as well as between each form element and a multiple-label concept nested within it, FOCIH generates either an *n*-ary relationship set or a set of binary relationship sets. If the multiple-label element is the only element in the form or the only element nested under another form element, FOCIH generates a set of binary relationship sets between the form-title concept and each of the concepts in the multiple-label element; otherwise FOCIH generates an *n*-ary relationship set. Thus, FOCIH generates an *n*-ary relationship set among *Country*, *Population*, and *Year* since the *Population-Year* element does not stand by itself as the only form element in the *Country* form. Our running example does not illustrate the case of a multiple-label form element by itself with no other form elements. As an example consider a form whose title is *Country* and whose only form element is a multiple-label element with the labels *Name*, *Capital*, *Population (2005 est.)*, and *Size (sq. km.)*. The rows in the multiple-label field would be various country names along with their capitals, populations, and sizes. In this case, FOCIH would generate four functional binary relationship sets: from *Country* to *Name*, from *Country* to *Capital*, from *Country* to *Population (2005 est.)*, and from *Country* to *Size (sq. km.)*.

Choice form elements. FOCIH generates a non-functional binary relationship set between the form-title concept and a top-level choice form element. Thus FOCIH generates a non-functional binary relationship set between *Country* and *Life Expectancy* as Figure 2 shows. For both mutually-exclusive and non-exclusive

---

[2] Lines connecting concepts denote relationship sets. Arrowheads on lines denote functional relationship sets from tail-side concepts to head-side concepts.

choice elements, FOCIH generates a generalization/specialization[3] (an is-a relationship among concepts) with the header label as the generalization concept and each of the labels on the choice list as specialization concepts. For Figure 1, FOCIH therefore generates a generalization/specialization with *Life Expectancy* as the generalization and *Male Life Expectancy* and *Female Life Expectancy* as specializations. Nesting choice form elements within choice elements extends the generalization/specialization hierarchy. Header labels of of nested generalizations must match upper-level specialization labels. We could, for example, extend the hierarchy by nesting *Male Life Expectancy 40-60* and *Male Life Expectancy 60+* under the upper-level specialization *Male Life Expectancy*. FOCIH imposes no constraints on generalization/specialization for non-exclusive form elements. For mutually-exclusive form elements, FOCIH adds a plus symbol to the triangle to designate the mutual exclusion. This, however, would be inappropriate for our example because we know that as life-expectancy values are harvested, some male and female life-expectancy values may be the same—thus, the male and female values are not mutually exclusive.

Although FOCIH is able to generate all concepts, relationship sets, and generalization/specialization hierarchies, it can generate only some of the constraints that may be desirable. FOCIH knows that relationship-set constraints from parent content to child concept should be functional when the child concept is a single-label/single-value element. From a form specification alone, however, FOCIH is not able to determine whether the inverse direction of a binary relationship set is functional. Names of countries, for example, might be unique and therefore functionally determine countries. In these cases, FOCIH initially imposes no constraints. Thus, in Figure 2, the *Name-Country* relationship set is not bijective. FOCIH, however, can later modify constraints based on observations as it harvests information from source documents. The non-mandatory constraints on the three relationship sets in Figure 2 appear because FOCIH observes that the first page from which it harvests information (i.e., the page in Figure 1) has no *Geographic Coordinate*, no *Water* area, and no *Land* area.

4   Automatic Semantic Annotation

Although users fill in a form manually, they only need to do this once for a single page from a site like the web site for the page in Figure 1 in which each of the many pages is machine-generated. To harvest and annotate information from the remaining pages, FOCIH determines the layout pattern for instance values in the first page and uses these patterns to extract instance values from remaining pages. To succeed FOCIH must (1) identify paths in HTML DOM trees leading to nodes that contain instance values and (2) identify the substrings in DOM-tree nodes that represent the instance values.

---

[3] A triangle denotes generalization/specialization with the apex of the triangle connected to the generalization and the base of the triangle connected to the specializations. Union ($\cup$), mutual-exclusion ($+$), or partition ($\uplus$) symbols may appear in the triangle to constrain the generalization/specialization.

Machine-generated web pages are *sibling pages*–pages with the same regular structure. Thus, we can usually locate corresponding DOM-tree nodes by following the same XPath from root to node. While harvesting information, FOCIH may encounter minor variations in the XPaths. If so, it adjusts by recording the variations and then searching for DOM-tree nodes in the remaining pages with any of the node's XPath variants.

A user-highlighted value can be the entire DOM-tree node (e.g, "Prague" in Figure 1) or a proper subpart of the string that constitutes the DOM-tree node (e.g., just the populated value in Figure 1).[4] For proper substrings within a node, FOCIH needs to know how to find the correct subpart within a DOM-tree node. Moreover, since a value can be composed of one or more highlighted values from one or more DOM-tree nodes (e.g., when longitude and latitude are in separate DOM-tree nodes), FOCIH needs to know how to compose values from different substrings of different nodes from the source page.

Considering these possibilities, we observe that there are two kinds of patterns: (1) individual patterns for entire strings, proper substrings, and string components, and (2) list patterns. Particularly for list patterns, but also as context for individual patterns, FOCIH has a default list of delimiters: ",", ";", ":", "|", " /", "\", "(", ")", "[", "]", "{", "}", *sos* (start of string) and *eos* (end of string). FOCIH also has a library of regular-expression recognizers for values in common formats, such as numbers, numbers with commas, decimal numbers, positive/negative integers, percentages, dates, times, and currencies [11].

- An *individual pattern* has left and right contexts and a regular-expression instance recognizer. For example, for the highlighted area value "78,866.00", the left context can be "\bsq\s*mi\s*" (word boundary with "sq" and "mi" surrounded by zero or more whitespace characters), the right context can be "\s*sq\s*km$" ("sq" and "km" surrounded by whitespace characters and then end of string), and the instance recognizer can be decimal number.
- A *list pattern* has a left context, a right context, a regular-expression instance recognizer, and a delimiter. The list of agriculture products in Figure 1 could have as its left context *sos*, as its right context *eos*, as its instance recognizer "\b([a-z]\s*)+\b" (any lower-case word or words), and as its delimiter "(,\s*)|(;\s*)" (either a comma-space or a semicolon-space).

We now explain how FOCIH detects and establishes patterns. FOCIH first determines whether a pattern is an individual pattern or a list pattern. Given a DOM-tree node and all its highlighted values, FOCIH groups the highlighted values that go to the same form entry together. If only one highlighted value from a node goes to a form entry, FOCIH establishes an individual pattern; and if several go to a form entry, FOCIH establishes a list pattern.

Secondly, for both individual and list patterns, FOCIH determines the context information. To determine the left or the right context of a highlighted value in a DOM-tree node, FOCIH initially takes the substring that is on the

---

[4] If an identified DOM-tree node is not already a string with no internal formatting tags, FOCIH removes the tags and converts the DOM-tree node to a simple string.

left or on the right of the highlighted substring until it reaches other highlighted values or the beginning or the end of the whole node string. FOCIH can further generalize the context in two ways. (1) if some of the context is recognizable as an instance of one of the regular-expression recognizers, FOCIH substitutes the recognized substring in the context by the recognizer. (2) FOCIH can generalize the context information when it sees more sibling-node contents during its harvesting phase of operation. Sometimes FOCIH cannot locate the context information in a newly encountered sibling page. This usually means that the initial context from the original sample page is too specific. FOCIH then tries to generalize the context by comparing context strings with the pattern and allowing non-delimiter characters that differ to be replaced by an expression that permits any characters.

Thirdly, for both individual and list patterns, FOCIH determines the regular expression pattern of the substrings of interest. If a highlighted substring can be recognized by a regular-expression recognizer in our library, FOCIH uses it as the instance recognizer for the pattern. If not, then the instance recognizer is an expression that recognizes any string. In this case, proper recognition depends on the left and right context, and for lists also the delimiter.

Finally, for list patterns, FOCIH compares the substrings between highlighted values to find delimiters. Looking particularly for delimiters in our list of delimiters, FOCIH attempts to identify a simple delimiter-separated list. It then constructs a regular expression for the delimiter. The agriculture list in Figure 1 is an example. For this list FOCIH creates the delimiter expression "[,;]\s*". For more complex cases such as the religions list in Figure 1, the list separator can include commentary or other values. In the religions list a percentage plus a comma and space separate the names of the religions, and the delimiter expression should be "\s*\d+(.\d+)?%,\s*". FOCIH generates this delimiter expression by (1) discovering that the percentage recognizer in the library recognizes part of every substring between highlighted values, (2) observing that a comma follows every percentage, and (3) noticing that the combination of the percentage and the comma covers the intervening substrings. In general, FOCIH checks library instance recognizers and standard delimiters to see if they cover intervening substrings; and when this is insufficient, FOCIH adds general character recognizers to cover the intervening substrings.

With path recognition and instance recognition in place, FOCIH can locate the information of interest from all the sibling pages in a site and appropriately associate each item of information with the generated ontology. FOCIH can thus semantically annotate each page in the site. In our implementation, FOCIH annotates each page and saves the annotated information in an RDF file. The information saved not only identifies each item of information and links it to a concept in the ontology, but also records its location on the page.[5] Thus, we

---

[5] We cache pages so that the annotations remain valid even when pages change. We can, of course, re-annotate pages as necessary.

are able to superimpose a web of data (the RDF files) over a web of pages and produce—at least as a research prototype—the envisioned Web 3.0.[6]

5   Experimental Results

FOCIH can always correctly generate ontologies according to user-created forms. How well FOCIH can automatically harvest and annotate information from sibling pages with respect to generated ontologies depends on how uniform the pages are. As an indication of what might be expected, we tested FOCIHs ability to do instance recognition by considering a number of different web pages.

We examined FOCIH's performance harvesting information from a collection of web pages about countries. For our experiment, we restricted our attention to 40 European country pages like the Czech Republic page in Figure 1. Starting with a human-created annotation for Germany, we ran FOCIH over the 40 pages, with the following results. For fields where the entire target node was the desired value (such as the country's official name or its capital), precision and recall were 100%. Several fields, such as the country's area or its population in a given year (the second of the population/year pairs in our test sample), required extraction from a proper subpart of the text of the target node. For the country's area, which was bounded on the left by the string "sq mi" and on the right by the string "sq km", precision and recall were 100%. For population as of a given year, precision was 100% for all values and recall ranged between 95% and 100%. But with a few additional annotation examples, recall rose to 100%.[7] Precision and recall were also 100% for lists of agricultural products. These 100% results are due to the regularity of the set of country pages.

As expected, the FOCIH prototype is less accurate on less regular elements. For example, the religions list exhibited significant variety from one page to the next. From our seed annotation of the Germany country page, the inferred list pattern was able to extract only about two thirds of the religion data correctly. When we added alternate annotation patterns, which FOCIH derived from other seed pages, precision rose to 95% while recall rose to 96%. A more sophisticated generalizing recognizer, which we are developing, should achieve even better precision and recall.

In principle, FOCIH is always able to achieve 100% precision and recall,[8] since the user can always fix every partial or incorrect annotation. FOCIH has three modes of operation: (1) fully automatic, (2) verify each annotation, and (3) verify only when FOCIH suspects it may be in error. When the tool operates

---

[6] For a full explanation of how we store the RDF files, link them to web pages, and query them either with SPARQL or our free-form query processor, see [31].

[7] In our current implementation, we have to restart FOCIH when giving additional annotation examples. We have not yet coded our prototype to generalize and make adjustments on the fly as it harvests.

[8] It is interesting to note that people themselves do not always agree, so we should not expect 100% from a machine; "100%" is only with respect to the person judging the results.

interactively (Modes 2 and 3), users may adjust the automatically extracted annotations to further train the harvester. Currently, our prototype implements only the first mode, but even now users can choose different initial pages and re-run the remaining pages to achieve effects similar to Modes 2 and 3.

In addition to the country pages sample, we applied FOCIH to web pages from the Gene Expression Omnibus site [14] and several e-commerce sites. The results in these cases were similar to results for country pages. FOCIH works well on pages that exhibit a high degree of regularity, and achieves less accuracy on pages or items within pages that are less regular.

An interesting avenue for future work that we discovered while annotating and harvesting e-commerce site pages is the interaction between HTML mark-up and the underlying text. Sometimes there is information we wish to extract in the mark-up itself (e.g., text in the "alt" attribute of image elements on the NewEgg.com site indicate the number user rating of a particular item). It would also be useful in several cases to take advantage of mark-up tags to delimit items in a list or to separate fields where one field is nested in the DOM tree within another field's node. For example, BarnesAndNoble.com embeds authors in an "em" element nested within an "h1" element representing the book title. Also, it is common to hyperlink items in a list, and thus the "a" tag structure could help parse list items. We are considering ways to generalize our annotation tool to allow annotation of mark-up text, and we are also working on a more robust implementation of FOCIH that will take advantage of these opportunities.

6   Further Reduction of Labor-Intensive Tasks

FOCIH helps users who do not know conceptual modeling to create ontologies and harvest and annotate information with respect to these ontologies. We want this process to be convenient with as much of the burden as possible shifted to the system. We see two major opportunities to further reduce labor-intensive tasks: (1) automatic initial form creation and (2) automatic initial form fill-in.

Often tables are "mirror images" of forms. When they are and when they only use FOCIH-equivalent layout structures, we can immediately generate FOCIH forms for them. As an example, consider the table in Figure 3. The nesting is the same as the nesting allowed in FOCIH forms. For example, the nesting of the single-label/single-value elements *Genetic Position*, *Genomic Position*, and *Genomic Environs* under *Location* in Figure 3 is identical to the nesting of *Water*, *Land*, and *Total* under *Area* in Figure 1. Isomorphic variations are acceptable, such as the nested table under *IDs* that has only one row, which we can consider as a simple layout variation of a group of single-label/single-value elements rather than a multiple-label/multiple-value element. With allowance for this variation, an analysis of the table in Figure 3 yields the form in Figure 4. A user can then modify the form,[9] if desired, and use it to harvest information.

---

[9] The icons in the form in Figure 4 allow users to label form fields (✎), to delete form fields (✗), or to add form fields either as additional top-level elements or elements nested within other elements (⬚ ▦ ▦ ▦ ▦).

Fig. 3. A Sample Table from WormBase [36].

We have implemented this reverse-engineering of tables into FOCIH forms based on a system called TISP (Table Interpretation for Sibling Pages) [29, 30]. TISP converts tables from sites like hidden-web sites that have machine-generated sibling pages into FOCIH forms and thus into FOCIH-generated ontologies. (Indeed, we generated the FOCIH form in Figure 4 with this implemented system.) Other table-interpretation systems (e.g., [13, 16, 25]) could also be used as front-end processors for generating FOCIH forms. Moreover, tables are not the only front-end structures from which we can derive forms. We have implemented a transformation algorithm to convert OWL ontologies to OSM ontologies and another algorithm to convert XML-Schema specifications to OSM ontologies. We have yet to implement an algorithm to convert OSM ontologies to FOCIH forms, but the process is reasonably straightforward given our algorithm that translates OSM ontologies to nested scheme trees [1, 20].

Besides generating an ontology, our TISP-to-FOCIH implementation also automatically harvests and annotates the data in the original table—indeed in all the sibling tables from a site (e.g., in the table in Figure 3 and all the sibling tables of the from the WormBase site). Thus, the system can also fill in the forms, and there is nothing for a user to do assuming the user is satisfied with the ontology automatically constructed by the TISP-to-FOCIH implementation. However, to facilitate the initial form-filling process for a form obtained in another way—perhaps by reverse-engineering an OWL ontology to a form—we

Fig. 4. Generated Form for Table in Figure 3 (Partial)

need an extraction ontology [11]. If we have an extraction ontology for the application, the system may be able to entirely, or at least partially, fill-in the form for the first page. If we do not have an extraction ontology for the application, after FOCIH harvests information from one web site for the application, we have many sample values for each concept in the ontology. These sample values are enough to enable FOCIH to begin to construct an extraction ontology. Thus, for a subsequent site in the same domain, FOCIH would likely be able to automatically initialize a form with some of its values extracted from a page. A user may need to add additional values and perhaps correct some values that may have been erroneously extracted. For each new site, FOCIH adds to the knowledge of the extraction ontology, and thus "learns" as it harvests and annotates, making the extraction ontology increasingly better over time and thus also shifting the burden for annotating increasingly more from the user to the system.

7 Concluding Remarks

We have implemented FOCIH, a form-specification and information-harvesting tool. FOCIH lets users who are not an experts in conceptual modeling or in ontology languages create an ontology and semantically annotate web pages with respect to the created ontology. We are able to guarantee that any user who can specify an ordinary form and can cut-and-paste values from web pages can successfully create an ontology and annotate web pages. Our implementation philosophy, however, is to shift as much of the the burden of ontology creation and semantic annotation to FOCIH as we can. Thus, we provide for: (1) automatic harvesting of information from the sibling pages of an initial annotated

page; (2) automatic creation of FOCIH forms and corresponding ontologies by reverse engineering structured documents such as tables, database schemas, OWL ontologies, or XML-schema specifications; (3) automatic initial form fill-in via extraction ontologies; and (4) semi-automatic creation of extraction ontologies.

Experience using FOCIH and experimental results are encouraging. Running the FOCIH prototype over dozens of pages on multiple sites shows that automatic harvesting performs well. The prototype often achieves near-perfect information harvesting for well-structured elements, which appear to be fairly common. More work needs to be done in processing sites with less regular structure, but the results achieved so far indicate that we can generalize our prototype implementation to cover less-regular pages. As for automatic creation of FOCIH forms, our implementation via TISP works well. And, as for our use of extraction ontologies with FOCIH, we still need to integrate our implementations and make them work synergistically. In the past, we have experimented extensively with extraction ontologies and have been able to achieve high precision and recall results for the domains we have studied (e.g., see [11]), so we are hopeful that the integration will bring about the expected synergy resulting in an even greater shift of the workload to the machine.

As FOCIH harvests information of interest, it semantically annotates the pages from which it extracts information and generates RDF data files. Hence, in the larger system in which FOCIH is embedded, the data of interest from a web site becomes accessible through a standard query interface. Queries yield not only direct answers, but for each retrieved data value also yield links back to the page from which the data was extracted. All of this points toward enabling the Web 3.0 vision—the superimposition of a web of data over a web of pages.

References

1. R. Al-Kamha. *Conceptual XML for Systems Analysis*. PhD dissertation, Brigham Young University, Department of Computer Science, June 2007.
2. L. Arlotta, V. Crescenzi, G. Mecca, and P. Merialdo. Automatic annotation of data extracted from large web sites. In *Proceedings of the Sixth International Workshop on the Web and Databases (WebDB 2003)*, pages 7–12, San Diego, California, June 2003.
3. F. Baader and W. Nutt. Basic description logics. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors, *The Description Logic Handbook*, chapter 2, pages 43–95. Cambridge University Press, Cambridge, UK, 2003.
4. M. Banko, M.J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 2670–2676, Hyderabad, India, January 2007.
5. S.M. Benslimane, M. Malki, M.K. Rahmouni, and D. Benslimane. Extracting personalised ontology from data-intensive web application: an HTML forms-based reverse engineering approach. *Informatica*, 18(4):11–534, 2007.
6. P. Buitelaar, D. Olejnik, and M. Sintek. OntoLT: A Protégé plug-in for ontology extraction from text based on linguistic analysis. In *Proceedings of the First Euro-*

pean Semantic Web Symposium (ESWS'04), pages 31–44, Heraklion, Greece, May 2004.

7. E. Chu, A. Baid, T. Chen, A. Doan, and J.F. Naughton. A relational approach to incrementally extracting and querying structure in unstructured data. In *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB'07)*, pages 1045–1056, Vienna, Austria, September 2007.

8. P. Cimiano, G. Ladwig, and S. Staab. Gimme' the context: context-driven automatic semantic annotation with C-PANKOW. In *Proceedings of the 14th International World Wide Web Conference (WWW205)*, pages 332–341, Chiba, Japan, May 2005.

9. P. Cimiano and J. Völker. Text2Onto—a framework for ontology learning and data-driven change discovery. In *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB'05)*, pages 227–238, Alicante, Spain, June 2005.

10. S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, K.S. McCurley, S. Rajagopalan, A. Tomkins, J.A. Tomlin, and J.Y. Zien. A case for automated large scale semantic annotations. *Journal of Web Semantics*, 1(1):115–132, December 2003.

11. D.W. Embley, D.M. Campbell, Y.S. Jiang, S.W. Liddle, D.W. Lonsdale, Y.-K. Ng, and R.D. Smith. Conceptual-model-based data extraction from multiple-record web pages. *Data & Knowledge Engineering*, 31(3):227–251, November 1999.

12. A. Gal, A. Anaby-Tavor, A. Trombetta, and D. Montesi. A framework for modeling and evaluating automatic semantic reconciliation. *The VLDB Journal*, 14(1):50–67, 2005.

13. W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krüpl, and B. Pollak. Towards domain-independent information extraction from web tables. In *Proceedings of the Sixteenth International World Wide Web Conference (WWW2007)*, pages 71–80, Banff, Alberta, Canada, May 2007.

14. Gene expression omnibus. http://www.ncbi.nlm.nih.gov/geo/, 2009.

15. S. Handschuh and S. Staab. Authoring and annotation of web pages in CREAM. In *Proceedings of the 11th International World Wide Conference (WWW2002)*, pages 462–473, Honolulu, Hawaii, May 2002.

16. P. Jha and G. Nagy. Wang notation tool: Layout independent representation of tables. In *Proceedings of the 19th International Conference on Pattern Recognition (ICPR08)*, Tampa, Florida, December 2008.

17. A. Kiryakov, B. Popov, I. Terziev, D. Manov, and D. Ognyanoff. Semantic annotation, indexing, and retrieval. *Journal of Web Sematics*, 2(1):49–79, December 2004.

18. M. Laclavík, M. Šeleng, E. Gatial, Z. Balogh, and L. Hluchý. Ontology based text annotation – OnTeA. In M. Duzi, H. Jaakkola, Y. Kiyoki, and H. Kangassalo, editors, *Proceedings of Information Modelling and Knowledge Bases XVIII, Frontiers in Artificial Intelligence and Applications*, volume 154, pages 311–315, Amsterdam, The Netherlands, 2007. IOS Press.

19. M. Michelson and C.A. Knoblock. Unsupervised information extraction from unstructured, ungrammatical data sources on the world wide web. *International Journal of Document Analysis and Recognition*, 10(3–4):211–226, 2007.

20. W.Y. Mok and D.W. Embley. Generating compact redundancy-free XML documents from concptual-model hypergraphs. *IEEE Transactions on Knowledge and Data Engineering*, 18(8):1082–1096, August 2006.

21. S. Mukherjee, G. Yang, and I.V. Ramakrishnan. Automatic annotation of content-rich html documents: Structural and semantic analysis. In *Second International Semantic Web Conference (ISWC 2003)*, pages 533–549, Sanibel Island, Florida, October 2003.

22. R. Navigli, P. Velardi, A. Cucchiarelli, and F. Neri. Quantitative and qualitative evaluation of the OntoLearn ontology learning system. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 1043–1050, Geneva, Switzerland, August 2004.

23. N.F. Noy, M. Sintek, S. Decker, M. Crubezy, R.W. Fergerson, and M. Musen. Creating semantic web contents with Protègè-2000. *IEEE Intelligent Systems*, 16(2):60–71, March–April 2001.

24. A. Pivk. Automatic ontology generation from web tabular structures. *AI Communications*, 19(1):83–85, 2006.

25. A. Pivk, Y. Sure, P. Cimiano, M. Gams, V. Rajkovič, and R. Studer. Transforming arbitrary tables into logical form with TARTAR. *Data & Knowledge Engineering*, 60:567–595, 2007.

26. S. Sarawagi. Information extraction. *Foundations and Trends in Databases*, 1(3):261–377, 2008.

27. W. Shen, P. DeRose, R. McCann, A. Doan, and R. Ramakrishnan. Toward best-effort information extraction. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1031–1042, Vancouver, British Columbia, Canada, June 2008.

28. P. Spyns, D. Oberle, R. Volz, J. Zheng, M. Jarrar, Y. Sure, R. Studer, and R. Meersman. OntoWeb—a semantic web community portal. In *Proceedings of the 5th International Conference on Practical Aspects of Knowledge Management (PAKM'02)*, pages 189–200, Vienna, Austria, December 2002.

29. C. Tao and D.W. Embley. Automatic hidden-web table interpretation by sibling page comparison. In *Proceedings of the 26th International Conference on Conceptual Modeling*, pages 556–581, Auckland, New Zealand, November 2007.

30. C. Tao and D.W. Embley. Automatic hidden-web table interpretation, conceptualization, and semantic annotation. *Data & Knowledge Engineering*, 2009. in press.

31. C. Tao, D.W. Embley, and S.W.Liddle. Enabling a web of knowledge. Technical report, Brigham Young University, 2009. (submitted for publication—draft manuscript available at deg.byu.edu).

32. Y.A. Tijerino, M. Al-Muhammed, and D.W. Embley. Toward a flexible human-agent collaboration framework with mediating domain ontologies for the semantic web. In *Proceedings of the ISWC-04 Workshop on Meaning Coordination and Negotiation*, pages 131–142, Hiroshima, Japan, November 2004.

33. M. Vargas-Vera, E. Motta, J. Domingue, M. Lanzoni, A. Stutt, and F. Ciravegna. MnM: Ontology driven tool for semantic markup. In *Proceedings of the Workshop Semantic Authoring, Annotation & Knowledge Markup (SAAKM 2002)*, Lyon, France, July 2002.

34. OWL Web Ontology Language Reference Manual. www.w3.org/TR/owl-ref. W3C (World Wide Web Consortium).

35. Y. Wang, J. Völker, and P. Haase. Towards semi-automatic ontology building supported by large-scale knowledge acquisition. In *AAAI Fall Symposium On Semantic Web for Collaborative Knowledge Acquisition*, volume FS-06-06, pages 70–77, Arlington, Virginia, October 2006.

36. Worm base! http://www.wormbase.org, 2005.