Ontology-driven Information Extraction with OntoSyphon

Luke K. McDowell[1] and Michael Cafarella[2]

[1] Computer Science Department, U.S. Naval Academy,
Annapolis MD 21402 USA. lmcdowel@usna.edu
[2] Dept. of Computer Science and Engineering, University of Washington,
Seattle WA 98195 USA. mjc@cs.washington.edu

Abstract. The Semantic Web's need for machine understandable content has led researchers to attempt to automatically acquire such content from a number of sources, including the web. To date, such research has focused on "document-driven" systems that individually process a small set of documents, annotating each with respect to a given ontology. This paper introduces OntoSyphon, an alternative that strives to more fully leverage existing ontological content while scaling to extract comparatively shallow content from millions of documents. OntoSyphon operates in an "ontology-driven" manner: taking any ontology as input, OntoSyphon uses the ontology to specify web searches that identify possible semantic instances, relations, and taxonomic information. Redundancy in the web, together with information from the ontology, is then used to automatically verify these candidate instances and relations, enabling OntoSyphon to operate in a fully automated, unsupervised manner. A prototype of OntoSyphon is fully implemented and we present experimental results that demonstrate substantial instance learning in a variety of domains based on independently constructed ontologies. We also introduce new methods for improving instance verification, and demonstrate that they improve upon previously known techniques.

1 Introduction

The success of the Semantic Web critically depends upon the existence of a sufficient amount of high-quality, relevant semantic content. But to date relatively little such content has emerged. In response, researchers have investigated systems to assist users with producing (or annotating) such content, as well as systems for automatically extracting semantic content from existing unstructured data sources such as web pages.

Most systems for automated content generation work as follows. Given a small to moderate size set of hopefully relevant documents, the system sequentially processes each document. For each document, the system tries to extract relevant information and encode it using the predicates and classes of a given ontology. This extraction might utilize a domain-specific wrapper, constructed by hand [1] or via machine learning techniques [2]. More recent domain-independent

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2006		**00-00-2006 to 00-00-2006**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Ontology-driven Information Extraction with OntoSyphon	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **U.S. Naval Aademy,Computer Science Department ,572 M Holloway Road,Annapolis,MD,21402**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Fifth International Semantic Web Conference (ISWC 2006), Nov 5-9 2006, Athens, GA

14. ABSTRACT

The Semantic Web's need for machine understandable content has led researchers to attempt to automatically acquire such content from a number of sources, including the web. To date, such research has focused on "document-driven" systems that individually process a small set of documents, annotating each with respect to a given ontology. This paper introduces OntoSyphon, an alternative that strives to more fully leverage existing ontological content while scaling to extract comparatively shallow content from millions of documents. OntoSyphon operates in an "ontology-driven" manner: taking any ontology as input, OntoSyphon uses the ontology to specify web searches that identify possible semantic instances, relations, and taxonomic information. Redundancy in the web, together with information from the ontology, is then used to automatically verify these candidate instances and relations, enabling OntoSyphon to operate in a fully automated, unsupervised manner. A prototype of OntoSyphon is fully implemented and we present experimental results that demonstrate substantial instance learning in a variety of domains based on independently constructed ontologies. We also introduce new methods for improving instance verification, and demonstrate that they improve upon previously known techniques.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**16**	

approaches have utilized a named entity recognizer to identify interesting terms, then used web searches to try to determine the term's class [3]. In either case, these are *document-driven* systems whose workflow follows the documents.

This paper describes OntoSyphon, an alternative *ontology-driven* information extraction (IE) system. Instead of sequentially handling documents, OntoSyphon processes the ontology in some order. For each ontological class or property, OntoSyphon searches a large corpus for instances and relations than can be extracted. In the simplest case, for instance, a `Mammal` class in the ontology causes our system to search the web for phrases like "mammals such as" in order to identify instances (and subclasses) of `Mammal`. We then use redundancy in the web and information in the ontology to verify the candidate instances, subclasses, and relations that were found. In this paper, we focus on learning instances.

Compared to more traditional document-driven IE, OntoSyphon's ontology-driven IE extracts relatively shallow information from a very large corpus of documents, instead of performing more exhaustive (and expensive) processing of a small set of documents. Hence, the approaches are complementary, and real world systems may profitably utilize both. We note, however, several benefits of ontology-driven IE. First, driving the entire IE process directly from the ontology presents a very natural path for exploiting all kinds of ontological data, e.g., utilizing class labels and synonyms for broader searching and exploiting instances and stated restrictions for verifying candidate facts. Second, a search-based system enables us to consider a much larger set of documents than could be handled via individual, document-driven processing. Only a small fraction of the corpus will be used for any one system execution, but much more potentially relevant information is accessible. Finally, ontology-driven IE can be easily focused on the desired results. Rather than processing all content from some documents and then looking for the desired info, we can instruct the system to search directly for relevant classes.

Our contributions are as follows. First, we introduce the ontology-driven paradigm for information extraction and explain its benefits compared to complementary approaches. Second, we explain how to apply this general paradigm to find instances from the web and demonstrate successful instance population for three different, independently created ontologies. Third, we evaluate several different techniques for improving the accuracy of instance identification and classification. In addition, we introduce two simple but highly effective improvements to previously known assessment techniques for such extractions. These improvements relate to adding or improving upon frequency-based normalization, and can be used even in contexts without an explicit ontology. Finally, we describe techniques for further improving accuracy based on explicitly leveraging the structure of the ontology.

The next section summarizes related work in this area. Section 3 summarizes OntoSyphon's operation, while Section 4 describes our methodology and evaluation metrics. Section 5 describes the existing and new techniques that we use for the key problem of assessing candidate instances. Finally, Section 6 presents experimental results, Section 7 discusses our findings, and Section 8 concludes.

	Text-based	Ontology-based	
		Document-driven	Ontology-driven
Domain-specific	Crystal[4], Citeseer, Opine[5]	WebKB[6], TAP[1], OntoMiner[7], OntoSophie[8], Armadillo[2], ADEL[9]	Cyc "web population"[10, 11], van Hage et al.[12]
Domain-independent	MindNet[13], Snowball[14], Cederberg et al.[15], KnowItAll[16], Pantel et al.[17]	Hahn et al.[18], S-CREAM[19], SemTag[20], KIM[21], PANKOW[3],	**OntoSyphon**

Table 1. A summary of work that attempts to (semi-)automatically extract instance-like content from the web or other text corpora. Note that an ontology-based system almost always utilizes a domain-specific ontology, but may still be a domain-independent *system* if it can easily exploit input ontologies from many different domains.

2 Related work on Information Extraction from the Web

The general task we face is to learn information from some textual source, such as the WWW, and encode that information in a structured language such as RDF. Table 1 provides an interpretation of the most relevant other work in this area. The rows of this table distinguish systems that are domain-independent from those that rely on domain-specific techniques or extraction patterns.

The columns of Table 1 explain the extent to which each system utilizes an explicit ontology. In the leftmost column ("Text-based") are information extraction systems that are not explicitly based on an ontology. For instance, Citeseer automatically extracts metadata about research publications, Opine [5] focuses on product reviews, and Crystal [4] uses a domain-specific lexicon to learn text extraction rules by example. Amongst more domain-independent systems, MindNet [13] builds a semantic network based on dictionary and encyclopedia entries, while Snowball [14] learns relations (such as `headquartersOf`) based on an initial set of examples. KnowItAll [16] learns instances and other relations from the web. Many such systems [15–17] learn hyponym or is-a relationships based on searching for particular lexical patterns like "cities such as ...," inspired by Hearst's original use of such patterns [22]. Our work uses these same patterns as building blocks, but exploits an ontology to guide the extraction and assessment, and to formally structure the results.

Some of these text-based systems, such as MindNet, use their input corpus to derive an ontology-like structured output. In contrast, we call a system *ontology-based* if it specifies its output in terms of a pre-existing, formal ontology. These systems almost always use a domain-specific ontology in their operation, but we consider a system to be domain-independent if it can operate without modification on ontologies covering a wide range of domains.

The majority of these ontology-based systems are document-driven: starting from a particular document (or set of documents), they try to annotate all of the entities in that document relative to the target ontology. For instance, TAP [1] exploits a variety of wrappers to extract information about authors, actors, movies, etc. from specifically identified websites such as Amazon.com. We-

bKB [6] and Armadillo [2] both use supervised techniques to extract information from computer science department websites. Amongst more domain-independent systems, SemTag [20] and KIM [21] scan documents looking for entities corresponding to instances in their input ontology. Likewise, S-CREAM [19] uses machine learning techniques to annotate a particular document with respect to its ontology, given a set of annotated examples. PANKOW [3] annotates a specified document by extracting named entities from the document and querying Google with ontology-based Hearst phrases. For instance, if the entity "South Africa" is found in a document, PANKOW would issues multiples queries like "South Africa is a river" and use hit count results to determine which ontology term (river, country, etc.) was the best match. These systems all use an ontology to specify their output, but make limited use of information that is contained in the ontology beyond the names of classes and properties that may be relevant.

OntoSyphon offers a complementary approach of being ontology-based and *ontology-driven*. Instead of trying to learn all possible information about a particular document, we focus on particular parts of an ontology and try to learn all possible information about those ontological concepts from the web. In addition, we seek to use ontological data and structure to enhance our assessment of the content that is found (see Section 6).

The only work of which we are aware that adopts a somewhat similar approach is that of Matuszek et al. [10, 11] and van Hage et al. [12]. Both systems use an ontology to generate web search terms, though neither identifies this ontology-driven approach or examines its merits. van Hage et al. use the searches to find mappings between two given ontologies, whereas Matuszek et al. use the searches to identify instances and relations that could be inserted into the (large) Cyc ontology. Matuszek et al. use more sophisticated natural language processing than we do, and use the existing Cyc ontology to perform more kinds of reasoning. Compared to OntoSyphon, however, the systems of van Hage and Matuszek perform much less accurate verification of content learned from the web, either assuming that a human will perform the final verification [10] or treating all web candidates as correct because of data sparsity [12]. In addition, both systems only discover information about instances or classes that are already present in their ontology, and both are domain-specific. Matuszek's system, for instance, depends upon manually generated search phrases for a few hundred carefully chosen properties.

Ontology learning systems seek to learn or extend an ontology based on examination of a particular relevant corpus [23–26]. Some such systems [24–26] use Hearst-like patterns to identify possible subclass relations. Ontology learning systems, however, presume a particularly relevant corpus and do not focus on learning instances (with some limited document-driven exceptions, e.g., Text2Onto [26]). In addition, the goal of producing a very accurate ontology leads to very different verification techniques, usually including human guidance and/or final verification. OntoSyphon instead operates in a fully automatic, unsupervised manner, and uses the web rather than require that a domain-specific corpus be identified.

Init: SearchSet = {R} + O.subclassesOf(R)
SearchSet = {Animal} + {Amphibian, Arthropod, Bird, Fish,...}

1. C = PickAndRemoveClass (SearchSet)
C = `Bird`

2. Phrases = ApplyPatterns(C)
Phrases = {"birds such as ...", "birds including ...", "birds especially ...",
"... and other birds", "... or other birds"}

3. Candidates += FindInstancesFromWeb (Phrases)
Candidates = {..., (kookaburra, Bird, 20), (oriole, Bird, 37), ... }

4. If MoreUsefulWork(SearchSet, Candidates), goto Step 1

5. Results = Assess (O, Candidates)

(kookaburra, Bird, 20)		*Results* = {	
(kookaburra, Mammal, 1)		(kookaburra, Bird, 0.93),	*LA: 1.00*
(leather, Animal, 1)	$\Longrightarrow$	(leather, Animal, 0.01),	*LA: 0.00*
(oriole, Bird, 37)		(oriole, Bird, 0.93),	*LA: 1.00*
(wildebeest, Animal, 56)		(wildebeest, Animal, 0.91)	*LA: 0.67*
(wildebeest, Mammal, 6)		}	

Fig. 1. OntoSyphon's algorithm (bold lines), given a root class R, for populating an ontology O with instances, and partial sample output (other lines). The text (oriole, Bird, 37) describes a candidate instance that was extracted 37 times. Step 5 converts these counts into a confidence score or a probability, and chooses the most likely class for candidates that had more than one possible class (results shown computed via Urns, see Section 5). "LA" is the "Learning Accuracy" of the final pair (see Section 4).

This paper focuses on demonstrating how a domain-independent, ontology-driven system can reliably extract instances using a few simple techniques. Overall performance could be increased even more by incorporating other techniques such as domain-specific pattern learning [14, 27, 16], automatic subclass identification [16], non-pattern based extraction [18, 24, 25, 23], and the combination of multiple sources of evidence [28].

3 Overview of OntoSyphon's Operation

Figure 1 gives pseudocode for OntoSyphon's operation. The input to OntoSyphon is an ontology O and a root class R such as `Animal`. The search set is initialized to hold the root term R and all subclasses of R. OntoSyphon then performs the following steps: pick a "promising" class C from the ontology (step 1), instantiate several lexical phrases to extract instances of that class from the web (steps 2-3), then repeat until a termination condition is met (step 4). Finally, use the ontology and statistics obtained during the extraction to assess the probability of each candidate instance (step 5). Below we explain in more detail.

1. **Identify a Promising Class:** OntoSyphon must decide where to focus its limited resources. For our initial experiments, we pragmatically chose to completely explore all subclasses of the user-provided root class. Future

work should consider how best to use OntoSyphon's limited resources when broader explorations are desired. For instance, we might like to chose the class that we know the least about (fewest instances), or instead focus attention on classes that are similar to those that yielded good results in the past. Finally, note that some classes (e.g., zip codes) may produce very large amounts of data that is accurate but uninteresting.

2. **Generate Phrases:** Given a class C, we search for lexico-syntactic phrases that indicate likely instances of C. For instance, phrases like "birds such as" are likely to be followed by instances of the class `Bird`. We use the 5 Hearst phrase templates [22] listed in the sample output of Figure 1. To generate the phrases, we use heuristic processing to convert class IDs such as `SweetDessert` to the search label "sweet desserts." Where present we also exploit alternative class labels that can be inferred from the ontology, e.g., through the definition of an equivalent class.
3. **Search and extract:** Next, we search the web for occurrences of these phrases and extract candidate instances. This could be done by submitting the phrases as queries to a search engine, then downloading the result pages and performing extraction on them. For efficiency, we instead use the Binding Engine (BE) [29]. BE accepts queries like "birds such as <NounPhrase>" and returns all possible fillers for the <NounPhrase> term in about a minute, but for only a 90-million page fragment of the web.
4. **Repeat** (for this paper, until *SearchSet* is empty).
5. **Assess Candidate Instances** (see Section 5).

We focus in this paper on basic instance learning, but this algorithm naturally lends itself to several future enhancements. For instance, in step 3, the candidate instances that are discovered will also discover subclasses. Such subclasses might be added to the *SearchSet* and/or might be used to extend the ontology itself. Our initial experiments have shown that, as is to be expected, such steps will increase recall but at some cost of precision. The next section discusses how we grade discovered subclasses for this work; future work will more fully investigate the benefits of exploiting these candidate subclasses.

4 Methodology

We ran OntoSyphon over the three ontologies shown in Table 2. All three ontologies were created by individuals not associated with OntoSyphon, and were freely available on the web. For each, we selected a prominent class to be the "root class," thereby defining three different domains for evaluation: Animals, Food, and Artists. Note that instances for the first two domains are dominated by common nouns (horses, sushi), whereas the latter yields mostly proper nouns (Michelangelo). These choices encompass a variety of domains and ontology types. For instance, the Animal ontology was fairly complete but shallow, while the Artist ontology covers artists in general but most classes focus on musical artists. The Food ontology has been used for demonstrating OWL concepts; it contains more complex constructions and classes such as `NonSpicyRedMeat`.

Domain	Ontology used	# Subs.	Avg. Depth	# Graded
Animals	sweet.jpl.nasa.gov/ontology/biosphere.owl	28	1.04 (max 2)	300
Artists	www.kanzaki.com/ns/music.rdf	39	2.63 (max 4)	940
Food	www.w3.org/TR/owl-guide/food.rdf	31	2.13 (max 3)	170

Table 2. The domains and ontologies used for our experiments. The third column gives the number of subclasses of the chosen root term, followed by the average (and maximum) depth of these subclasses relative to the root term. The last column is the number of candidates that were human-graded for evaluation (5% of the total found).

OntoSyphon operates in a totally unsupervised manner and outputs a ranked list of candidate instances for the ontology. Because there is no accurate, authoritative source for determining the full, correct set of instances for our three domains, we cannot report recall as an absolute percentage, and instead report just the number of distinct, correct instances found. In addition, we must evaluate correctness by hand. To do this, we created a "gold standard" as follows: all system configurations produce the same basic set of candidate instances for a given ontology. A human evaluator (one of the authors) classified a random sample of 5% of this set (see Table 2). For each candidate, the evaluator chose the best, most specific ontology class available, while allowing for multiple senses. So a `dog` would be marked as a `Mammal` rather than the less specific `Animal`, while `Franz Liszt` would be marked as a `Composer`, `Pianist`, and `Conductor`. Two classes, `ExoticSpecies` and `IndigenousSpecies`, were removed from Animals because they were too subjective for a human to evaluate. To reduce bias, the evaluator had no knowledge of what class OntoSyphon assigned to each candidate nor OntoSyphon's assigned probability for that candidate.

Candidates with no correct class for that domain (e.g., truck) were marked as incorrect, as were misspelled or incomplete terms (e.g., the artist "John"). To decide whether a candidate was a proper instance or a subclass, we assumed that the ontology was fairly complete and tried to follow the intent of the ontology. Candidates that could be properly classified as an instance of a leaf node in the ontology were treated as instances. Candidates that were already present in the ontology as a class or that seemed to parallel an existing class (e.g., the discovered "fiddler" and the existing class `Pianist`) were counted as incorrect. Finally, candidates that did not fit the intent of the ontology were marked incorrect. For instance, we considered the `Animal` ontology to be about types of animals (dogs, cats), so specific animals like "King Kong" or "Fido" were incorrect; other animal ontologies might make different decisions (see Section 7).

This evaluation produced the function $gold_O()$, where $gold_O(i)$ is the set of classes assigned to candidate instance i by the evaluator for ontology O. Then, given a candidate instance i and a class c, we define the pair (i, c) to be:

- **correct** if $c \in gold_O(i)$,
- **sensible** if $\exists c' \in gold_O(i)$ s.t. $c' \in subclasses(c)$,
- or **incorrect** otherwise.

For instance, if $gold_O(dog) = \{Mammal\}$, then $(dog, Mammal)$ is correct, $(dog, Animal)$ is sensible, and $(dog, Reptile)$ is incorrect.

Let X be the output of the system for some experimental condition, where X consists of a set of pairs of the form (i, c), and where each candidate instance i appears in only one pair.[3] Then the *recall* is the number of pairs in X that are correct, and the *precision* is the fraction of pairs in X that are correct. These metrics are useful, but count only instances that were assigned to the most correct class possible, and thus do not fully reflect the informational content of the result. Consequently, we primarily report our results using the *sensible-recall*, which is the number of pairs in X that are sensible. In addition, we follow the example of several others in using *learning accuracy (LA)* instead of exact precision. The LA measures how close each candidate pair (i, c) was to the gold standard $(i, gold_O(i))$. This measurement is averaged over all pairs to yield a precision-like number ranging from zero to one where $LA(X) = 1$ indicates that all candidate pairs were completely correct.

We follow the general definition of Learning Accuracy from Cimiano et al. [3], which requires the least common superconcept (lcs) of two classes a and b for ontology O:

$$lcs_O(a, b) = \arg\min_{c \in O}(\delta(a, c) + \delta(b, c) + \delta(top, c)) \tag{1}$$

where $\delta(a, b)$ is the number of edges on the shortest path between a and b. Given this definition, the taxonomic similarity $Tsim$ between two classes is:

$$Tsim_O(d, e) = \frac{\delta(top, f) + 1}{\delta(top, f) + 1 + \delta(d, f) + \delta(e, f)} \tag{2}$$

where $f = lcs_O(d, e)$. We then define the average learning accuracy for ontology O of a set X of candidate pairs as:

$$LA_X = \frac{1}{|X|} \sum_{(i,c) \in X} max(0, \max_{c' \in gold(i)} Tsim_O(c, c')) \tag{3}$$

5 Assessing Candidate Instances

Extracting candidate instances from the web is a noisy process. Incorrect instances may be extracted for many reasons including noun phrase segmentation errors, incorrect or incomplete sentence parsing, or factual errors in the web corpus. Because OntoSyphon operates in an unsupervised manner, it is thus critical to be able to automatically assign a confidence value to the instances that are produced. These values can be then used to expunge instances that are below some confidence threshold and/or to provide reliability estimates to applications that later make use of the output data.

[3] OntoSyphon assigns only a single class to each instance, which is often sufficient but restrictive for domains like Artists. Future work should consider more general techniques.

Below we describe the five assessment techniques that we consider in our initial results. Each is used to assign a confidence score or probability to a candidate pair (i, c). In what follows, $count(i, c, p)$ is the number of times that the pair (i, c) was extracted using the pattern p, and $hits(y)$ is the number of hits for the term y alone in the corpus.

1. Strength: Intuitively, if the pair $(dog, Mammal)$ is extracted many times from our web corpus, this redundancy gives more confidence that that pair is correct. The Strength metric thus counts the number of times a candidate pair was observed across all extraction patterns P:

$$Score_{strength}(i, c) = \sum_{p \in P} count(i, c, p) \tag{4}$$

This metric was also used by PANKOW [3], although the counts were obtained in a different manner.

2. Str-Norm: The Strength metric is biased towards instances that appear very frequently on the web. To compensate, Str-Norm normalizes the pattern count by the number of hits for the instance alone:

$$Score_{str-norm}(i, c) = \frac{\sum_{p \in P} count(i, c, p)}{hits(i)} \tag{5}$$

Similar normalization techniques are found in many systems (e.g., [16, 28]).

3. Str-Norm-Thresh: The normalization performed by Str-Norm can be misleading when the candidate instance is a very rare term or a misspelling. Consequently, we created a modified Str-Norm where the normalization factor is constrained to have at least some minimum value. We found that a variety of such thresholds worked well. For this work we sort the instances by $hits(i)$ and then select Hit_{25}, the hit count that occurs at the 25th percentile:

$$Score_{str-norm-thresh}(i, c) = \frac{\sum_{p \in P} count(i, c, p)}{max(hits(i), Hit_{25})} \tag{6}$$

4. Urns: OntoSyphon, like some other systems, extracts candidate facts by examining a large number of web pages (though we use the aforementioned Binding Engine to perform this process very efficiently). Prior work has developed the Urns model to apply in such cases [30, 31] and has shown it to produce significantly more accurate probabilities than previous methods such as PMI (pointwise mutual information) or noisy-or. The Urns model treats each extraction event as a draw of a single labeled ball from one or more urns, with replacement. Each urn contains both correct labels (from the set C), and incorrect labels (from the set E); where each label may be repeated on a different number of balls. For instance, $num(C)$ is the multi-set giving the number of balls for each label $i \in C$. Urns is designed to answer the following question: given that a candidate i was extracted k times in a set of n draws from the urn (i.e., in n extractions from the corpus), what is the probability that i is a correct instance? For a single urn, if s is the total number of balls in the urn, then this probability is computed as follows:

$$P(i \in C | NumExtractions_i(n) = k) = \frac{\sum_{r \in num(C)} \frac{r}{s}^k (1 - \frac{r}{s})^{n-k}}{\sum_{r' \in num(C \cup E)} \frac{r'}{s}^k (1 - \frac{r'}{s})^{n-k}} \quad (7)$$

Urns operates in two phases. In the first phase, the set of all candidate extractions is used to estimate the needed model parameters ($num(C)$ and $num(C \cup E)$), using Expectation Maximization (EM). In particular, $num(C)$ is estimated by assuming that the frequency of correct extractions is Zipf-distributed, and then estimating the exponent z which parameterizes this distribution. In the second phase, a single pass is made over the extractions and each is assigned a probability using the estimated model parameters and an integral approximation to Equation 7 [30].

Urns was designed to assign probabilities to a set of extractions that were targeting a single class, and the EM phase relies upon having a sufficient number of samples (roughly 500) for estimation. In our context, very few classes yield this many extractions on their own. Fortunately, we found that combining the candidates from all of the classes of a single ontology together for model estimation yielded good results, better than performing per-class model fitting with prior parameters for low-sample classes.

5. Urns-Norm: The Urns model, like Strength, does not exploit the frequency with which a candidate appears in the corpus. Instead, each instance is assumed to occur with equal probability anywhere along the aforementioned Zipf distribution. Introducing an appropriate prior probability of an input candidate's location along this curve would improve accuracy, but would significantly complicate the model and computation.

Fortunately, we can approximate the benefit of such a change with a much simpler approach. We begin by sorting the input data set X by $hits(i)$. We then run EM both on the "lower" half (which contains the less frequent terms), obtaining parameter z_L and on the whole data set, obtaining the aforementioned z. We then compute a parameter z_i for each instance i as follows:

$$z_i = max(z_L, z + log(hits(i)) - \sum_{(j,c) \in X} \frac{log(hits(j))}{|X|}) \quad (8)$$

The probability for candidate (i, c) is then computed using z_i. Intuitively, the log functions increase z_i when a candidate i is more frequent than average, thus forcing i to have more pattern matches to obtain a high probability. On the other hand, the max function ensures that very infrequent words (particularly misspellings) do not obtain an artificially high probability, by insisting that the minimum z_i is a value appropriate for the "lower" half of the inputs (z_L).

Strength, Str-Norm, and Urns have been used in some form in other work, though Urns required some adaption for our multi-class problem. Both Str-Norm-Thresh and Urns-Norm, however, are novel contributions of this work that we found to be very effective, and that should also provide significant improvements in other systems.

6 Experimental Evaluation

In this section we experimentally evaluate OntoSyphon. We first consider its overall performance and the impact of different assessment techniques, then examine how to further improve accuracy by exploiting ontological structure.

6.1 Assessing Instances

Figures 2-4 show the results of executing OntoSyphon on our sample ontologies. Each line represents one output of the system using a particular assessment technique. To create one point on the line, we chose a threshold and then removed from the output set all candidate pairs whose assigned confidence values were below that threshold. The x-axis measures the sensible-recall of this modified set, and the y-axis shows the LA of this set. Varying the threshold thus produces a line with properties similar to a classical precision/recall tradeoff curve.

The data demonstrates that OntoSyphon was able to find a substantial number of sensible instances for all 3 domains. In addition, the data shows that some of our tested assessment techniques are quite effective at identifying the more reliable instances. In particular, the techniques that perform normalization (Str-Norm, Str-Norm-Thresh, and Urns-Norm) show consistent improvements over both techniques that do not (Strength and Urns). Consider, for instance, the "50% recall" point, where each technique has a sensible-recall equal to half of the maximum achieved under any situation (e.g., where sensible-recall equals 900 for Animals). At this point, Urns-Norm and Str-Norm-Thresh increase LA compared to the no-normalization techniques by 82-110% for Animals, 19% for Artists, and 51-56% for Food.

Overall, both Urns-Norm and Str-Norm-Thresh perform consistently well. Str-Norm also performs well, except that it has many false positives at low recall. It gets fooled by many incorrect terms that are very infrequent on the web, and thus have a high score after normalization, even though they were extracted only once or a few times. For instance, Str-Norm incorrectly gives a high score to the misspelled terms "mosquities" for Animals, "Andrea Cotez" for Artists, and "pototato" for Food.

OntoSyphon found the most sensible instances for Artists, but our assessment techniques worked least well on this domain, showing a fairly flat curve. One reason is that the assessment techniques are fooled by a large number of spurious candidates that come from one ambiguous class, `Players`. This class (intended for musical instrument players), finds mostly sports team participants (e.g. "Greg Maddux") and a few digital music software products ("WinAmp"). Figure 5 shows the results if this class is removed from OntoSyphon's search process (Section 7 describes how this could be done automatically). With Str-Norm-Thresh, LA increases from 0.49 to 0.64 at 50% recall.

A more fundamental problem is the smaller amount of redundancy in our corpus for the Artists domain. For instance, a sensible instance for Animals is extracted by OntoSyphon's patterns on average 10.9 times vs. 2.1 times for an incorrect instance (23.9 vs 3.5 hits for Food). However, Artists, even with `Player`

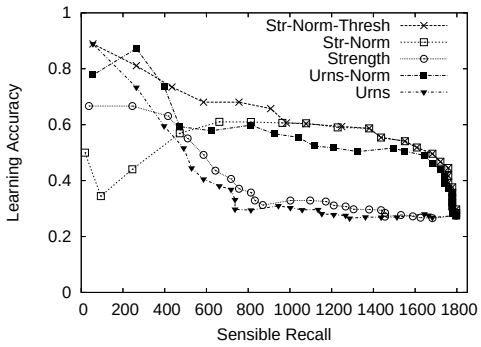

Fig. 2. Animal Domain

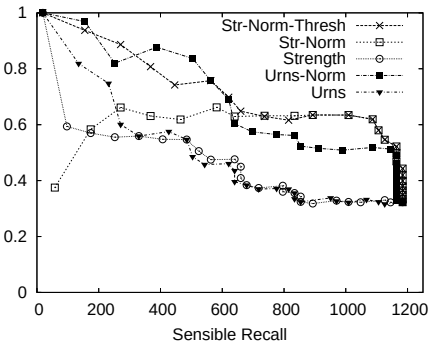

Fig. 3. Food Domain

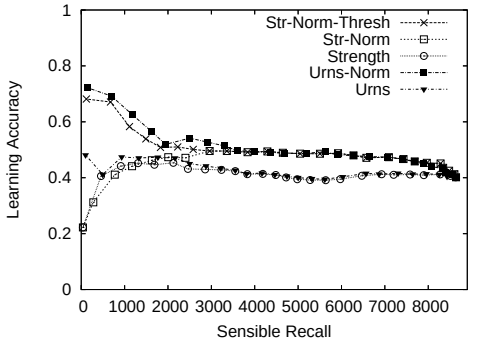

Fig. 4. Artist Domain

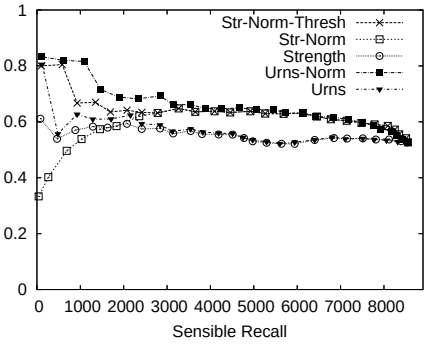

Fig. 5. Artist (without `Player`)

removed, averages only 3.0 hits for sensible instances vs. 2.1 hits for incorrect instances. This smaller split yields a much more difficult assessment task, and additional work is needed to more fully exploit the potential of such domains.

6.2 Leveraging Ontological Structure

When confronted with multiple possible classes for a candidate instance (e.g., is "lemur" an `Animal`, `Mammal`, or `Reptile`?), the results above chose the instance/class pair with the highest score. We also tried normalizing the metrics above by class frequency to influence this choice of classes, but we found that this produced erratic results without improving accuracy. Instead, we found better results by more explicitly leveraging the ontology to pick the best class via an average score computed over all classes. The component that each class contributes is weighted by class frequency and by the taxonomic similarity $Tsim$:[4]

$$Class(i) = \arg\max_{c \in O} \sum_{c' \in O} Score(i, c') \frac{\dot{Tsim}_O(c, c')}{hits(c')} \tag{9}$$

[4] Maedche et al. previously used a similar, non-normalized technique ("tree ascent") for ontology learning, with some mixed success [24].

For our corpus and domains, only about 25% of distinct instance terms were extracted for more than one subclass. Thus, this enhancement had a very small effect on overall results. However, looking only at that 25% of the results where a class decision must be made, we found that this new technique consistently improved LA for both Animals and Artists, and had negligible effect for Food. For instance, for Str-Norm-Thresh at 50% recall, LA improved from 0.88 to 0.95 for Animals and from 0.62 to 0.76 for Artists.

Thus, exploiting taxonomic relationships in the ontology can improve accuracy, though the small number of possible classes found for each instance limits the impact of this technique. There remains room for improvement, because even with this enhancement in the full results only 44-69% (average 60%) of the instances that were assigned a valid class by the evaluator were assigned that same fully correct class by the system. In the future, we would like to explore combining this technique with web-wide statistics computed via pointwise mutual information (PMI) [16]. Using PMI would require a significant number of additional search queries, and would need to be expanded to deal with our multi-class scenario. However, this approach should boost performance by providing additional supporting data while still enabling us to leverage our efficient gathering of candidate instances from our base 90 million page corpus.

7 Discussion & Future Work

Overall, we conclude that OntoSyphon was highly effective at extracting instances from a web corpus, and that our new assessment techniques (Str-Norm-Thresh and Urns-Norm) significantly improved the accuracy of the results. In particular, using Str-Norm-Thresh OntoSyphon was able to achieve a LA of about 0.6 while extracting 1400 sensible Animal instances (78% of the total found), 1100 Food instances (93% of the total), and (after removing `Player`) 7500 Artist instances (87% of the total). Even higher accuracy may be obtained for Animals and Food at a cost of reduced recall.

A Learning Accuracy of 0.6 is on par with the results surveyed by Cimiano et al. [3]. They report LA of between 0.44 and 0.76 (with an average of 0.61 for independent systems) and recall ranging from 0.17 to 0.31 (with an average of 0.24). These results are not directly comparable with ours, since these systems perform a different task (annotating individual documents rather populating a ontology from many documents), use different ontologies, and in some cases evaluate LA using only those terms marked by the evaluator as valid for some class.[5] Also, these systems generally define recall as the percentage of results from the complete gold standard that was found by the system. For our open-web system, however, recall is reported as a raw number or as a percentage

[5] For instance, C-PANKOW [3] appears to compute LA using only instances that were assigned a class by both the system and an evaluator. For our system (see Equation 3) it seemed more accurate to instead assign a value of zero to a pair (i, c) produced by the system but for which $gold_O(i) = \emptyset$ (e.g. for $(truck, Animal)$). This decision lowers our LA values in comparison.

of the set of all answers found by any execution of the system (as with [16]). Nonetheless, the magnitude of these previous results demonstrate that an LA of 0.6 is reasonable, and our results show that OntoSyphon can find many instances at this accuracy level.

Normalization was essential to our results. Such normalization is particularly important for OntoSyphon, as opposed to document-driven systems, because the candidate terms are not guaranteed to come from domain-relevant, somewhat reliable input documents. These same factors caused us to achieve the best, most consistent results only when that normalization was constrained by a minimum threshold to account for very rare or misspelled words.

Our two metrics that performed such normalization, Str-Norm-Thresh and Urns-Norm, both performed well and about comparably. These results are consistent with earlier, non-normalized findings: while Urns was found to be greatly superior to many other techniques in terms of producing accurate probabilities [30], simple Strength-like measures performed almost as well if only a relative confidence ranking, not a probability was required [31]. Because Urns and Urns-Norm are more complex to compute, many situations may thus call for using the simpler Str-Norm-Thresh. On the other hand, users of the final, populated ontology may find actual probabilities very helpful for further processing, in which case Urns-Norm may be best.

Finally, OntoSyphon in its present form is clearly not suited for populating every kind of ontology. For instance, ontologies describing things or events that are mentioned only a handful of times on the web are not well suited to our current strategy of using simple pattern-based extractions followed by redundancy-based assessment. Likewise, classes that are either complex (`NonBlandFish`) or ambiguous (`Player`) will not yield good results. We intend to develop techniques to address these issues, for instance, by recognizing ambiguous classes by the small degree of overlap between a class and its parent (as is the case with `Player` and `Artist`) or by adding additional search terms to disambiguate such classes during extraction. Lastly, deciding whether a term such as "dog" should be a subclass or an instance can be challenging even for human ontologists. More work is needed to help OntoSyphon honor the intent of an ontology, e.g., by considering subclasses and instances already present in that ontology.

8 Conclusion

The Semantic Web critically needs a base of structured content to power its applications. Because of the great variety of information, no one approach will provide everything that is needed. Much content can only be created by human annotators, and incentives are needed to motivate this work. Other data is contained in documents that can be effectively leveraged via the document-driven approaches described in Section 2. This paper has focused on an alternative ontology-driven method to extract large amounts of comparatively shallow information from millions of web pages. This approach lets us leverage the existing work of skilled ontology designers, extract information from a very large corpus, and focus extraction efforts where it is most valuable and relevant.

While additional work is needed to demonstrate that OntoSyphon is robust across an even wider range of ontologies and can extract non-instance information, our initial results have demonstrated the feasibility of OntoSyphon's ontology-driven, domain-independent approach. We successfully extracted a large number of instances from a variety of independently-created ontologies. We demonstrated how different assessment techniques affect the accuracy of the output, and introduced simple improvements to existing assessment techniques that significantly improved upon these results. Because these techniques, Str-Norm-Thresh and Urns-Norm, are easy to implement modifications to techniques that have been used for other tasks, our improvements should carry over easily to many other systems (e.g., [3, 16, 10, 17, 12]). Future work will examine the many promising directions for further improvements in this area.

Acknowledgements. Thanks to Christopher Brown, Martin Carlisle, Frederick Crabbe, Oren Etzioni, Jeff Heflin, and the anonymous referees for their helpful comments on aspects of this work. This work was partially supported by the Naval Research Council, ONR grants N0001405WR20153 & N00014-02-1-0324, NSF grant IIS-0312988, DARPA contract NBCHD030010, as well as gifts from Google, and carried out in part at the University of Washington's Turing Center.

References

1. Guha, R., McCool, R., Miller, E.: Semantic search. In: World Wide Web. (2003)
2. Chapman, S., Dingli, A., Ciravegna, F.: Armadillo: harvesting information for the semantic web. In: Proc. of the 27th Annual Int. ACM SIGIR conference on Research and development in information retrieval. (2004)
3. Cimiano, P., Ladwig, G., Staab, S.: Gimme' the context: Context-driven automatic semantic annotation with C-PANKOW. In: Proc. of the Fourteenth Int. WWW Conference. (2005)
4. Soderland, S.: Learning to extract text-based information from the World Wide Web. In: Knowledge Discovery and Data Mining. (1997) 251–254
5. Popescu, A.M., Etzioni, O.: Extracting product features and opinions from reviews. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP). (2005)
6. Craven, M., DiPasquo, D., Freitag, D., McCallum, A.K., Mitchell, T.M., Nigam, K., Slattery, S.: Learning to construct knowledge bases from the World Wide Web. Artificial Intelligence **118**(1/2) (2000) 69–113
7. Davalcu, H., Vadrevu, S., Nagarajan, S.: OntoMiner: Bootstrapping and populating ontologies from domain specific web sites. IEEE Intelligent Systems **18**(5) (2003) 24–33
8. Celjuska, D., Vargas-Vera, M.: Ontosophie: A semi-automatic system for ontology population from text. In: International Conference on Natural Language Processing (ICON). (2004)
9. Lerman, K., Gazen, C., Minton, S., Knoblock, C.A.: Populating the semantic web. In: Proceedings of the AAAI 2004 Workshop on Advances in Text Extraction and Mining. (2004)
10. Matuszek, C., Witbrock, M., Kahlert, R., Cabral, J., Schneider, D., Shah, P., Lenat, D.: Searching for common sense: Populating cyc from the web. In: Proc. of AAAI. (2005)

11. Schneider, D., Matuszek, C., Shah, P., Kahlert, R., Baxter, D., Cabral, J., Witbrock, M., Lenat, D.: Gathering and managing facts for intelligence analysis. In: Proceedings of the International Conference on Intelligence Analysis. (2005)
12. van Hage, W., Katrenko, S., Schreiber, G.: A method to combine linguistic ontology-mapping techniques. In: Fourth International Semantic Web Conference (ISWC). (2005)
13. Richardson, S., Dolan, W., Vanderwende, L.: Mindnet: acquiring and structuring semantic information from text. In: COLING. (1998)
14. Agichtein, E., Gravano, L.: Snowball: Extracting relations from large plain-text collections. In: Proceedings of the Fifth ACM International Conference on Digital Libraries. (2000)
15. Cederberg, S., Widdows, D.: Using LSA and noun coordination information to improve the precision and recall of automatic hyponymy extraction. In: Seventh Conference on Computational Natural Language Learning (CoNLL). (2003)
16. Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A., Shaked, T., Soderland, S., Weld, D., Yates, A.: Unsupervised named-entity extraction from the web: An experimental study. Artificial Intelligence **165**(1) (2005) 91–134
17. Pantel, P., Ravichandran, D., Hovy, E.: Towards terascale knowledge acquisition. In: 20th International Conference on Computational Linguistics (COLING). (2004)
18. Hahn, U., Schnattinger, K.: Towards text knowledge engineering. In: AAAI/IAAI. (1998)
19. Handschuh, S., Staab, S., Ciravegna, F.: S-CREAM - semi-automatic creation of metadata. In: EKAW. (2002)
20. Dill, S., Eiron, N., Gibson, D., Gruhl, D., Guha, R.: Semtag and seeker: Bootstrapping the semantic web via automated semantic annotation. In: Proc. of the Twelth Int. WWW Conference. (2003)
21. Kiryakov, A., Popov, B., Terziev, I., Manov, D., Ognyanoff, D.: Semantic annotation, indexing, and retrieval. Journal of Web Semantics **2**(1) (2004) 49–79
22. Hearst, M.: Automatic acquisition of hyponyms from large text corpora. In: Proc. of the 14th Intl. Conf. on Computational Linguistics. (1992)
23. Cimiano, P., Hotho, A., Staab, S.: Learning concept hierarchies from text corpora using formal concept analysis. Journal of Artificial Intelligence Research **24** (2005) 305–339
24. Maedche, A., Pekar, V., Staab, S.: Ontology learning part one – on discovering taxonomic relations from the web. In: Web Intelligence, Springer (2002)
25. Alfonesca, E., Manandhar, S.: Improving an ontology refinement method with hyponymy patterns. In: Language Resources and Evaluation (LREC). (2002)
26. Cimiano, P., Volker, J.: Text2onto - a framework for ontology learning and data-driven change discovery. In: Int. Conf. on Applications of Natural Language to Information Systems. (2005)
27. Snow, R., Jurafsky, D., Ng, A.Y.: Learning syntactic patterns for automatic hypernym discovery. In: NIPS 17. (2004)
28. Cimiano, P., Pivk, A., Schmidt-Thieme, L., Staab, S.: Learning taxonomic relations from heterogeneous evidence. In: ECAI-2004 Workshop on Ontology Learning and Population. (2004)
29. Cafarella, M., Etzioni, O.: A search engine for natural language applications. In: Proc. of the Fourteenth Int. WWW Conference. (2005)
30. Downey, D., Etzioni, O., Soderland, S.: A probabilistic model of redundancy in information extraction. In: Proc. of IJCAI. (2005)
31. Cafarella, M., Downey, D., Soderland, S., Etzioni, O.: KnowItNow: fast, scalable information extraction from the web. In: Proc. of HLT-EMNLP. (2005)