Harvesting Domain Specific Ontologies from Text

Hamid Mousavi
CSD, UCLA
hmousavi@cs.ucla.edu

Deirdre Kerr
CRESST, UCLA
dkerr@cse.ucla.edu

Markus Iseli
CRESST, UCLA
iseli@cse.ucla.edu

Carlo Zaniolo
CSD, UCLA
zaniolo@cs.ucla.edu

***Abstract*—Ontologies are a vital component of most knowledge-based applications, including semantic web search, intelligent information integration, and natural language processing. In particular, we need effective tools for generating in-depth ontologies that achieve comprehensive converge of specific application domains of interest, while minimizing the time and cost of this process. Therefore we cannot rely on the manual or highly supervised approaches often used in the past, since they do not scale well. We instead propose a new approach that automatically generates domain-specific ontologies from a small corpus of documents using deep NLP-based text-mining. Starting from an initial small seed of domain concepts, our OntoHarvester system iteratively extracts ontological relations connecting existing concepts to other terms in the text, and adds strongly connected terms to the current ontology. As a result, OntoHarvester (i) remains focused on the application domain, (ii) is resistant to noise, and (iii) generates very comprehensive ontologies from modest-size document corpora. In fact, starting from a small seed, OntoHarvester produces ontologies that outperform both manually generated ontologies and ontologies generated by current techniques, even those that require very large well-focused data sets.**

I. Introduction

By introducing *concepts* and their *relations*, *ontologies* provide a critical and necessary information structure that facilitates the processes of sharing, reusing, and analyzing domain knowledge in Semantic Web and other knowledge-based systems [10]. This has given rise to ambitious systems, such as FreeBase [5], DBPedia [4], YaGo2 [12], and ProBase [36], that support general ontologies alongside their large-scale knowledge bases. However, in spite of their size and breadth, current systems do not provide comprehensive enough ontologies for most applications requiring domain-specific knowledge. Indeed, the need for more complete ontologies has motivated much research work in recent years.

Many existing domain-specific ontology generators use manual or highly supervised techniques [6][33][26][7][30] [35]. Although these approaches make it easier to incorporate knowledge from domain-experts, their high demands on time and resources impair their practical scalability. To address this problem, automatic or semi-automatic approaches have been proposed, using statistical techniques, occasionally combined with shallow NLP-based methods [20][29][3][28][8]. These automatic approaches have achieved a fair amount of success, but have also encountered several limitations. In fact, as stated in Poon et al. [28], none of the existing techniques have achieved a higher accuracy than 91%. Moreover to reach these relatively high levels they require large training data sets highly focused on the domains of interest. A simple analysis of these limitations suggests that these issues rise due to the inability of the mentioned approaches to take full advantage of the linguistics morphologies in the text. This observation has inspired the design of our *OntoHarvester* system, that uses deep NLP analysis to automatically generate domain-specific ontologies from unstructured text. In this paper, we describe this approach, and show that it outperforms previous approaches in terms of accuracy and coverage.

OntoHarvester starts with an initial ontology (*Seed*) and iteratively extends it with new terms carefully mined from the input text. At each iteration, OntoHarvester only accepts new terms that are semantically connected to the current concepts. In this way, OntoHarvester remains focused on the specified domains and achieves high resistance to noise. On the other hand, since the semantic connections are mined through a deep NLP-based technique, the system is able to generate very comprehensive ontologies from small corpora. This represents a major improvement over other automatic techniques, which require large corpora to generate domain-specific ontologies, as shown in detailed comparisons presented later in the paper.

The main tasks performed in each iteration by OntoHarverster can be summarized as follows:

1) Building TextGraphs: This task is performed by using our text-mining system *SemScape* [22][24]. SemScape uses a pattern-based technique, to capture *candidate terms* and their grammatical links in the text, and represents them as weighted hyper-graphs, called *TextGraphs*.

2) Extracting Ontological Relations: OntoHarvester uses graph patterns (*Graph Domain* or *GD* Rules) similar to those of Hearst's [11] to extract ontological relations (initially *part_of* and *type_of* relations).

3) Extracting Concepts: The ontological relations are used to detect candidate terms that are strongly connected to the current concepts in the ontology. These terms are then added to the current ontology.

4) Finding New Relation Types: In this step, OntoHarvester finds new ontological relation types between currently accepted concepts. These new relation types will help OntoHarvester find more related concepts in the next iterations.

More specifically, we present the following contributions:

- We propose OntoHarvester, and explain how it generates comprehensive domain-specific ontologies from very noisy corpora. New terms are recognized and incorporated into the ontology on the basis of (i) their

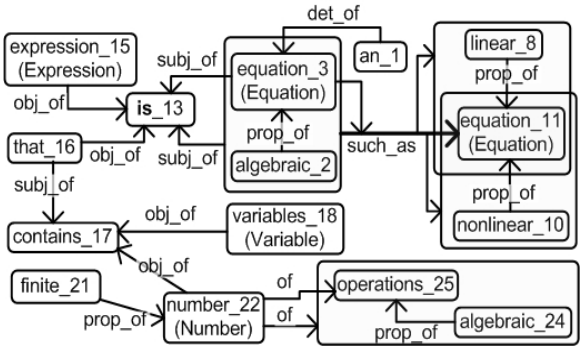

Fig. 1. Part of the TextGraph for our motivating example. For simplifying our discussion, we do not show the confidence of the links in this graph.

frequency and correctness confidence and (ii) the strength of their connections with existing concepts (An important difference from most of the existing works that only use (i)). The connections are generated using Graph-based patterns (*Graph Domain* or *GD* rules). This represents a significant improvement over existing pattern-based techniques that rely on tree-based patterns or regular expressions. The patterns are all available in [23].

- We present a novel technique to automatically suggest new possible domain-specific ontological relation types to improve the final Ontology. This is another feature differentiating OntoHarvester from existing works which mostly use predefined relation types.
- To provide a more clear insight on the quality of the generated ontologies, we use an application-focused evaluation technique to evaluate our system. To this end, we implement a "topic identifier" system that mainly uses an ontology to suggest topics for a given piece of text. The results indicate that the ontology created by OntoHarvester can significantly improve the performance of applications using the ontology. Moreover, extensive experiments to measure precision and recall on several application domains also show significant uniform improvements over previous approaches.

II. From Text to TextGraphs

Since OntoHarvester uses the *SemScape* text mining framework [22] [24], we next give a short overview of this framework. The ultimate goal of SemScape is to convert text into a machine-friendly weighted graph structure, called *TextGraph*, which contains grammatical links between terms and words in the text. With TextGraphs, more effective and efficient algorithms can be designed to extract knowledge from text by combining graph-based and statistical methods. To better understand this process, we use the following motivating sentence throughout the paper:

Example: "*An algebraic equation, such as a linear or nonlinear equation, is an expression that contains variables and a finite number of algebraic operations.*"

To generate TextGraphs, the following tasks are performed:
Task 1: After preprocessing the text and partitioning it into its sentences, SemScape parses the sentences with a probabilistic parser [2] and generates their parse trees (*PTs*). For each sentence, the framework considers more than one parse tree to improve the quality of the final TextGraphs.

Task 2: Using about 135 tree-based patterns (TD rules), SemScape annotates each node in the PTs with information referred to as *Main-Parts* (*MPs*). MPs carry up hidden information from the lower branches of PTs to the upper nodes during preprocessing time. Thus, MPs reduce and simplify the required higher-level mining patterns/rules, resulting in an overall faster text mining process. More specifically, the MP for noun phrase np in a parse tree contains all possible single- or multi-word terms (sometimes called *candidate terms* in the literature) which np may be representing. E.g., the noun MPs for "*linear and nonlinear equation*" are "*equation*", "*linear equation*", and "*nonlinear equation*".

Task 3: After annotating PTs, SemScape uses around 270 TD rules to generate grammatical links between words and *candidate terms*. For instance, we generate links such as <*equation*, *subject_of*, *is*>, <*algebraic equation*, *subject_of*, *is*>, <*linear*, *property_of*, *equation*>, etc. These links are also assigned a confidence value, based on SemScape's confidence on the correctness of the patterns generating them. By integrating these links, SemScape generates the TextGraphs such as the one depicted in Figure 1.

Task 4: SemScape also provides techniques for *Anaphora and Co-reference Resolution* to improve the TextGraphs by replacing pronouns and coreferences with the terms they are referring to [23] [24]. E.g., the pronoun "*that*" in the running example is resolved with "*expression*".

Since both sets of mentioned TD rules (in tasks 2 and 3) are syntactical, not semantic, they are domain-independent. As a result, they can be used as-is in descriptive documents for new domains. This claim is verified in Section IV, where we used 4 different domains to evaluate our system.

A very important feature of this framework is the ability to adopt an ontology and provide more related *candidate terms* with respect to that ontology. To understand why this is important, notice that for complex noun phrases (NPs) there might be several possible candidate terms. Not all of these candidate terms are useful or meaningful. Thus, suggesting all of them as MPs will lead to a very large set of candidate terms which lowers the efficiency. To prevent this problem, if the system is fed with an ontology, say O, SemScape only generates candidates terms that either i) contain less than three words, ii) are part of an existing concepts in O, or iii) contain a concept from O. In many cases this generates all possible candidate terms; however for many long noun phrases, this helps reduce the size of the TextGraphs.

As shown in Figure 1, all the candidate terms which are also a concept in O are tagged with their concepts in the TextGraphs. For instance, nodes *"expression"*, *"equation"*, *"number"*, and *"variables"* are in O and tagged with their corresponding concepts. Notice that the corresponding concept of a term may be a synonym or alias (e.g., *"variables"* is an alias for *"variable"*).

III. Ontology Generation

Once the TextGraphs (TGs) are created for the given corpus (τ), OntoHarvester starts its main discovery loop by iterating over the following three main steps.

- *Extracting Relations*: As explained in Section III-A, OntoHarvester extracts ontological relations between the existing terms in TGs at the beginning of each iteration. For instance, from the TextGraph of our running example, OntoHarvester is able to generate relations such as <*algebraic equation*, *type_of*, *expression*>, <*linear equation*, *type_of*, *equation*>, <*variables*, *part_of*, *expression*>, etc.
- *Extracting Concepts*: Next, OntoHarvester detects new concepts and aliases using those extracted relations that connect a concept in the current ontology (O) to a non-concept term in TGs (Sections III-B and III-C). In our running example, considering the above mentioned relations and the fact that "*equations*" is already a concept, OntoHarvester accepts "*linear equation*" (and similarly "*nonlinear equation*" and "*algebraic equation*") as new concepts.
- *Suggesting New Relation Types*: Before starting the next level, OntoHarvester extract new relation types using *semantic links* provided in TextGraphs (Section III-D). OntoHarvester initially starts with three types of ontological relations: $type\_of$, $part\_of$, and $rename$. Later at the end of each iteration, it finds new ontological relation types by considering frequently observed semantic links between current concepts in O.

The key idea in above steps is to extend the current ontology (O) only with terms highly related (connected) to O. In this way, as opposed to similar systems [34], OntoHarvester is able to stay focused and generate highly related concepts (to the specified domain) from a noisy corpus. The domain is essentially specified by some of its concept in the initial ontology, O_0. We should note that our experiments in Section IV indicate that O_0 does not need to be large at all. Usually a handful of initial concepts and a small corpus suffice for OntoHarvester to generate a well-focused ontology.

In theory, OntoHarvester stops its iterations once no new concept is found. However, iterations at the end of the tail normally produce very few new concepts. Thus, in practice, we stop the iterations either i) when a certain number of iterations are performed or ii) when the number of newly generated concepts are less than a predefined threshold (min_c).

A. Extracting Ontological Relations

To generate ontological relations between terms in the TextGraph of each sentence in τ, we use graph domain (GD) patterns/rules described next. Using a syntax similar to SPARQL [1], GD rules are used to specify and detect patterns in TextGraphs which indicate particular fragments of knowledge. In our case, we are interested in using GD rules in order to find taxonomic relations, namely *type_of* (*IS_A*) and *part_of* (*HAS_A*). For instance, consider the relation <*algebraic equation*, *type_of*, *expression*> in our running example. To extract such a link from our motivating example, the following GD rule is used:

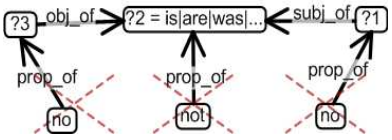

Fig. 2. Pattern Graph for Rule 1.

Rule 1.

```
SELECT ( ?1 "type_of" ?3 )
WHERE {
    ?1 "subj_of" ?2.
    ?3 "obj_of" ?2.
    NOT("not" "prop_of" ?2).
    NOT("no" "prop_of" ?1).
    NOT("no" "prop_of" ?3).
    FILTER (regex(?2, "^is^|^are^|^was^|^were^|^be*", "i")) }
```

As shown in Figure 2, Rule 1 specifies a pattern in which two nodes (labeled *?1* and *?3*) are connected through a third node (*?2*) that represents any tense of the verb "to be". The NOT parts filter out the negative sentences; This is usually a challenging task in most current systems. Rule 1 catches the following four results from the TextGraph in Figure 1:

- <*equation, type_of, expression*>,
- <*algebraic equation, type_of, expression*>,
- <*equation, type_of, that*>, and
- <*algebraic equation, type_of, that*>.

For the last two relations, OntoHarvester uses SemScape's *Anaphora Resolution* component to resolve the pronoun "*that*" and replace it with its resolved term (for this case "*expression*"). If no resolution is found, the last two relations will be rejected, since the term "*that*" is in our black list of concepts, alongside all other pronouns. For the above case, pronoun resolution results in re-generation of some of the exiting relations. However, Rule 2 shows how pronoun resolution can also generate new relations which most existing works are not able to capture.

Rule 2.

```
SELECT ( ?1 "part_of" ?3 )
WHERE {
    ?3 "subj_of" ?2.
    ?1 "obj_of" ?2.
    NOT("not" "prop_of" ?2).
    NOT("no" "prop_of" ?1).
    NOT("no" "prop_of" ?3).
    FILTER (regex(?2, "^include*|^contain*|^consist*|...", "i")) }
```

This rule has a very similar structure to Rule 1, but here it defines a pattern for subjects and objects connected with a verb indicating inclusion (e.g. *include*, *contain*, and *consist*). For our example this rule generates relations such as:

- <*variables, part_of, that*>
- <*finite number of algebraic operations, part_of, that*>

Now, by resolving "*that*" to "*expression*", OntoHarvester generates new relations such as:

- <*variables, part_of, expression*>
- <*finite number of algebraic operations, part_of, expression*>

In OntoHarvester, we have generated 48 GD rules for taxonomic relation extraction [23], and our experiments in Section IV show that these powerful rules can be used (with no changes) in different domains. Out of these 48 rules, 30 are for *type_of* relations and the rest are for *part_of*.

Combining the Relations: Each of the extracted relations from the GD rules introduced earlier are assigned with a correctness confidence. The confidence value of a relation generated from pattern p is set to the minimum confidence among the edges in the matching graph for patten p. After applying the GD rules over all TextGraphs, we combine the generated ontological relations from different TextGraphs. The combination process of relations confidence is very similar to that for TD rules; Since we need to keep the confidence values below 1, OntoHarvester uses the formula C=C_1+(1-C_1)$\times C_2$ to combine the confidence of two relations having C_1 and C_2 as their respective confidence values. It is easy to see that $1 \geq C \geq C_1$ and $1 \geq C \geq C_2$.

That is, each time we encounter new evidence for an existing relation, we increase its confidence in proportion to the new relation's confidence. Moreover, for each unique relation, we store its frequency, which measures the number of times this relation is extracted from different sentences.

Relations between concepts in O whose confidence and frequency exceed confidence threshold ($Conf$) and frequency threshold ($Freq$) are added into ontology O. Therefore, we will not include relations with high confidence but low frequency, inasmuch as insufficient evidence exists for them. Symmetrically, relations with high frequency and low confidence will also not be accepted.

B. Extracting Concepts

As mentioned previously, nodes in TextGraphs can be concepts, non-concept (or candidate) terms, and other words in the text such as verbs, articles, etc. Thus we need to ascertain which of the current candidate terms are actually new concepts. Most previous works rely on statistical or frequency-based techniques to accept candidate terms as concepts, and require large corpora to perform satisfactory. OntoHarvester instead exploits the ontological relations generated in the previous section to detect new concepts. The main intuition in OntoHarvester is that if a candidate term is a concept in the domain of τ, it should be connected to some other concepts of the domain through one or more relations. Thus, OntoHarvester accepts a term as a new concept if it has a "*strong connection*" to other existing concepts in O. Here, by a *strong connection*, we mean that the relations from the new term to existing concepts have frequency and confidence values that are respectively greater than $Freq$ and $Conf$.

With this intuition, all ontological relations connecting any concept in O to a candidate term, say CT_i, are retained and their confidence and frequency are combined exactly as described in the previous section. Now, CT_i's with high frequency and confidence are accepted as new concepts, providing that they do not contain: 1) a word from the black list (e.g. same, an, etc.), 2) an attributive adjective (e.g. short, similar, etc.), 3) a comparatives or superlative adjective, 4) a verb, an adverb, or a pronoun, and 5) a numeric or a symbol. After these adjustments, we accept those candidate terms whose confidence and frequency are larger than our pre-specified thresholds, and add them to O.

C. Extracting New Aliases

A single concept may have several "names". Research shows that people use different names for the same concept more than 80% of the time [9]. Abbreviations, variations of the term (e.g. plural form), acronyms, aliases, etc. may be used to address the concepts. To find these different names for same concepts, OntoHarvester uses heuristics explained next.

We first directly use either TD or GD rules to find aliases explicitly mentioned in the text. (e.g. *"a node may be called a vertex."*). This technique considers aliases as a new type of ontological relations as in Section III-A. Currently we have generated 29 TD rules and seven GD rules for finding such relations respectively in annotated parse trees and TextGraphs. If two concepts are "*strongly*" connected through *type_of* in both directions, they will be considered aliases of each other. (e.g. <*edge, type_of, side*> and <*side, type_of, edge*> means that "*side*" and "*edge*" are aliases.) We also use synonym information from WordNet and Wikipedia Redirect pages to detect aliases between concepts in the current ontology. Finally, adjustments similar to those discussed in the previous subsection will be applied to pick the final aliases. We group all the aliases of the same concept and (randomly) consider one of them as the head of the group. In the next iteration, at the concept annotation time, we use the head of each group for tagging any of the aliases in that group.

D. Extracting New Relation Types

SemScape is also able to generate general semantic links between the terms in the TGs [21][24]. The most prominent example of such semantic links is the <*subject*, *verb*, *object*> link, in which the *verb* in a sentence is connecting its *subject*s to its *object*s. Using these links, OntoHarvester detects new ontological relation types between exiting concepts. These new relation types will be used in the next iteration of the OntoHarvester to find more related concepts.

Let R_T be the set of current ontological relation types. R_T initially contains $type\_of$, $part\_of$, and *rename* relation types. The average number of relations in O for these relation types is called avg_{RT}. Also, let f_{sl} be the number of time the semantic link sl is generated between concepts in the current ontology O. Thus, for each sl, if f_{sl} is greater than $avg_c \times avg_{RT}$, OntoHarvester adds sl to the the current relation types (R_T). Where, avg_c is a constant factor for tuning the number of newly found relation types. Our experiments show that if $avg_c > 1$, OntoHarvester will find at most three new relation types (which are "*of*", "*be*", and "*and*"). It is also important to mention that the newly found relation types may not always resemble any specific ontological relation. Nevertheless, they are very useful for connecting concepts

TABLE I
ONTOHARVESTER (OH) VS. CRCTOL ON PGT DATA SET. ($Prob$=.98)

	Freq	Total	Precision	Recall	F-Score
Concepts-OH	4	1307	92.9%	41.0%	56.9%
Concepts-OH	5	1069	93.0%	35.5%	51.4%
Concepts-OH	6	925	93.1%	23.0%	36.9%
Concepts-CRCTOL	N/A	N/A	92.8%	4.1%	7.8%
Relations-OH	4	2587	83.8%	25.0%	38.5%

of the same domain. That is, they will help significantly in discovering new concepts for the specified domain.

IV. EXPERIMENTAL RESULTS

We evaluate OntoHarverster's performance using manual and automatic techniques in this section. All experiments are performed on a single machine with 16 cores of 2.27GHz and 16GB of main memory running an Ubuntu12. On average, our system spends 3.07 seconds for parsing each sentence on a single CPU (5.2 sentences per second). The data sets[1], used in our evaluations, are as follows:

- **SFF**: The first data set is actually Chapter 5 of the book entitled *"Naval Shiphandler's Guide"*. This data set, referred to as SFF (for Ship Fire Fighting), contains around 2,200 paragraphs and 5,000 sentences.
- **PGT**: The second data set is the PGT data set that was also used in [14]. It contains the reports on "Patterns of Global Terrorism" for years 1991-2002[2] for a total of about 3,000 paragraphs and 8,000 sentences.
- **MATH**: The third data set is much larger than the other two and contains 24,184 long abstracts (≈172K sentences) of mathematics related pages in Wikipedia. To generate this data set, referred to as MATH, we collected pages in Wikipedia for which one of their ancestor categories belongs to the *"mathematics"* category.
- **ANIM**: Similarly, a forth data set, called ANIM is created. This data set contains 11966 long abstracts (≈71K sentences) for pages having ancestor category "*animal*".

The last two data sets present challenging test cases since they are very noisy. For instance a careful analyses of the MATH data set shows that less than 45% of those pages describe concepts directly related to mathematics. In fact, several pages in this data set actually belong to related topics, such as physics, computer science, astronomy, book titles, universities, people names, etc. We should add that the GD patterns for relations extraction are created only once and independently from these data sets. For each data set, we also generated one or two initial ontologies (seeds) as follows:

- $O_{\texttt{Anim}}$: It contains: *Animal*, *Mammal*, *Fish*, *Bird*, and *Insect*.
- $O_{\texttt{sff\_short}}$: It contains: *AFFF (Aqueous Film Forming Foam)*, *Extinguisher*, *Fire(s)*,*Firefighting*, *Firemain*, *Hose*, *Nozzle*, *Party*, *Smoke*, *Ventilation*, and *Water*.
- $O_{\texttt{sff\_long}}$: It contains 150 concepts and 20 aliases. The index of the *"Naval Shiphandler's Guide"* book is used to create this ontology.

[1]Datasets and results are available at http://semscape.cs.ucla.edu

[2]Downloaded from the FSA: http://www.fas.org/irp/threat/terror.htm

TABLE II
THE IMPACT OF THE SEED SIZE FOR SFF.

	Seed	Total No.	Precision
Concepts	$O_{\texttt{sff\_small}}$	1131	90.0%
Concepts	$O_{\texttt{sff\_large}}$	1238	90.9%
Aliases	$O_{\texttt{sff\_small}}$	187	91.5%
Aliases	$O_{\texttt{sff\_large}}$	216	94.9%
Relations	$O_{\texttt{sff\_small}}$	2515	76.1%
Relations	$O_{\texttt{sff\_large}}$	2770	76.4%

- $O_{\texttt{pgt}}$: It contains: *Attack(s)*, *Bombing(s)*, *Hostage(s)*, *Incident(s)*, *Target(s)*, *Terrorism*, *Terrorist(s)*, *Terrorist group*, and *Threat(s)*.
- $O_{\texttt{Math}}$: It contains 877 concepts and 186 aliases which are manually extracted from Common Core State Standards for Mathematics as part of other projects[3].

A. OntoHarvester vs. CRCTOL

We first compare OntoHarvester's performance with CRCTOL [14]–one of the few works which uses a deep NLP-based approach to generate high-quality domain-specific ontologies. CRCTOL provides a natural test bed for comparison experiments, since it obtains the most accurate results among previous approaches and has both precision and recall computed on a publicly available data set (PGT). In order to compare OntoHarvester's performance with CRCTOL, we ran OntoHarvester for 10 iterations starting by $O_{\texttt{pgt}}$.

To measure the precision of our results, we manually graded 10% of the generated concepts and relations. As for recall, we found 200 concepts and 100 relations related to the global terrorism domain from the PGT documents for the year 1991. Then, we checked these concepts and relations against the ones OntoHarvester has generated to compute the coverage[4]. Table I depicts the comparative results for the extracted concepts of these two approaches. The recall and F-Score of OntoHarvester is significantly higher than those for CRCTOL, while the precision is slightly improved. This improvement is mainly due to the fact that, as opposed to CRCTOL and many similar techniques, OntoHarvester uses ontological relations to find more related concepts to the domain, and gradually extend the ontology. In this way, many wrong or unrelated concepts, which are frequently mentioned in the text, are eliminated since they are not semantically connected to the current Ontology. On the other hand, CRCTOL learns the concepts in a single phase and independently from the relations. Thus to avoid wrong concept and improve its accuracy, CRCTOL needs to increase the frequency threshold which consequently reduces the number of generated results. We expect similar improvement for the extracted relations, but unfortunately no result is reported for relations by CRCTOL.

B. The Impact of Seed's Size

To study the impact of seed's size, we ran OntoHarvester on SFF using the large and small seeds ($O_{\texttt{sff\_large}}$ and $O_{\texttt{sff\_small}}$) considering the same thresholds mentioned earlier.

[3]For more information visit: http://www.cse.ucla.edu/.

[4]Similar annotation is done for CRCTOL, but our efforts to contact the authors failed.

TABLE III
THE PRECISION/RECALL RESULTS OF RUNNING ONTOHARVESTER ON DATA SETS FROM DIFFERENT DOMAINS.

Data Set	Total	Precision	Recall	F-Score
Concepts-SFF-SmallSeed	1,131	90.0%	-	-
Concepts-SFF-LargeSeed	1,238	90.9%	-	-
Concepts-PGT	1,307	92.9%	41.0%	56.9%
Concepts-MATH	19,476	83.6%	51.7%	63.8%
Concepts-ANIM	6,595	82.3%	-	-
Relations-SFF-SmallSeed	2,515	76.1%	-	-
Relations-SFF-LargeSeed	2,770	76.4%	-	-
Relations-PGT	2,587	83.8%	25.0%	38.5%
Relations-MATH	26,814	87.4%	-	-
Relations-ANIM	2,987	81.2%	-	-

As shown in Table II, although the smaller seed contains only 10 concepts, rather than 150, starting from this much smaller seed, we were able to generate 91.3% of concepts one can generate by starting from the larger seed. Similar results hold for extracted links and aliases. Moreover, the accuracy of the results was not significantly affected. This simply verifies that users do not need to spend a lot of time creating the seed: A few domain-specific concepts and a small corpus are sufficient for OntoHarvester to provide high-quality ontologies.

C. OntoHarvester on Various Domains

We ran OntoHarvester for our four data sets and the precision and recall results are included in Table III. For all experiments, we ran 10 iterations and set the confidence threshold ($Conf$) to .98. As for the frequency threshold, we used $Freq$=4 for SFF and PGT, $Freq$=5 for ANIM, and $Freq$=6 for MATH to be proportional to the size of data sets.

Table III provides the precision of results obtained for various test sets. These results were derived by grading randomly selected samples with the help of four human graders. The samples' size was always 500 items or more. As shown in *Precision* column, OntoHarvester provides high-quality results for all four different domains. The provided accuracy for both concepts and relations in these cases is quite steady showing that OntoHarvester is general and domain-independent.

It is interesting to note the effect of noise in data sets on the precision of the generated concepts. MATH and ANIM that have more than 55% unrelated content are the ones with the lower precision. On the other end, PGT has the largest precision, since not only it is very well-focused but it also mentions repeatedly the same concepts and relations in different scenarios. The latter differentiates PGT from SFF data set. Our experiments also indicate that the quality of the created relations depends mostly on the complexity of the sentences in the text. For instance, since the sentences in SFF are more complex than sentences in the other three sets (SFF is written by a professional author as opposed to the others), extracted relations for SFF are of slightly lower accuracy.

We also evaluated the recall of the MATH experiment in a similar way to that for PGT. This time, we randomly graded 5% of Wikipedia titles (1219 titles) in the MATH data set and found 545 mathematic concepts among them. Then, we used these 545 concepts as a benchmark to compute the coverage of the results generated by OntoHarvester. As reported in

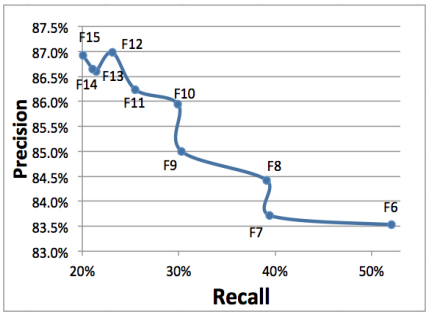

Fig. 3. The precision/recall comparison for the concepts generated for MATH. ($Freq$= 6, ..., 15, $Conf$=.99)

Table III, 51.7% of the benchmark's concepts are extracted by OntoHarvester as well. Noting that more than 39% of the benchmark concepts are mentioned only once in the entire data set, 9.5% of them have only one sentence as their abstracts, and several abstracts include formulas, symbols, or very formalized definitions, the estimated recall is actually quite satisfactory—statistical techniques are much less likely to find such low frequent concepts.

To study the effect of the frequency threshold ($Freq$) on the performance of OntoHarvester, we provided the precision and recall for the MATH experiments with frequency thresholds varying from 6 to 15. The results of this experiment are shown in Figure 3. Although, we can reach higher precision by increasing $Freq$, the recall drops quickly to less than its half. This is mainly due to the fact that many of the correctly extracted concepts are actually of very low frequency, and they are thus eliminated once $Freq$ is raised.

Finally, in Table IV, we provide the most frequent domain specific relation types suggested by OntoHarvester for each data sets. We excluded the generic relation types such as prepositions (*of, in, for, to, with, etc.*), conjunctions (*and* and *or*), and common verbs (*be* and *have*) from this table.

D. Application-Based Evaluation

Although previous experiments provide a good insight on the quality of the generated results, they are still subjective to the graders opinion. A more objective approach for evaluating ontologies is to use them in an application and evaluate the results of the applications. To this end, we apply the ontology generated by OntoHarvester to the problem of "Topic Identification" (TID) [31]. The goal in this problem is to identify the main topic(s) for a given text. Thus, we use the following algorithm, called TID, which is based on the technique proposed by Janik et al. [13]:

1) Constructing Initial Semantic Graph: For the given text

TABLE IV
DOMAIN SPECIFIC RELATION TYPES FOR EACH DATASET.

Dataset	Domain-Specific Relation types (ordered by frequency)
SFF	*explains, consistOf, include, beInvestigationOf, beInstalledIn, beProvidedBy, contain, beProvidedFor, provide, beLikelyIn*
PGT	*kill, beClaimedFor, arrest, beProvidedTo, relatedTo, beConductedBy, conduct, engagedIn, claim, sentencedTo*
MATH	*explains, beCalled, include, beUsed By, use, beUsedIn, beGivenBy, beDescribedBy, beRepresentedBy, knownAs*
ANIM	*include, explains, like, consistOf, contain, beFoundIn, referTo, liveIn, beCalled, beCauseBy, beSpeciesOf*

τ, we create all the semantic links in τ as explained in Section III-D. Using these semantic links, we then construct a semantic graph called G.

2) Pruning the Graph by Ontology: Based on the given nodes in an ontology, say O, we only keep nodes in G that are also in O. Each node, say n, is also assigned a weight (w_n) which is initiated by the degree (the number of connected links) of n in G. Later we use these weights to suggest final topics.

3) Populating the Graph by Taxonomy: We add taxonomical links to the remaining nodes (concepts) in G. This way more general terms and topics are added to the graph. Assume taxonomical link l=<n, $type\_of$, m> is used to extend the concept n in G. Thus, we add link l and node m (if it is not already there) to G and increment w_m by $c_d \times w_n$, where $c_d < 1$ is a constant decaying factor. For instance, if node "*Persian Cat*" is in the list, and O indicates "*Persian Cat*" is type of "*Cat*", we will add "*Cat*" to G as well. However, the weight of "*Cat*" is incremented only by a portion of the weight of "*Persian Cat*". This is because we do not want to give too much credit to more general topics so the final selected topics are specific enough for given text. Of course, if the text (τ) is mentioning other types of cats (e.g. Ragdoll, Munchkin, etc.) then the node "*Cat*" will receive more credit (weight) and its chance to be selected as the main topic increases. This step will be repeated for any newly added nodes to G.

4) Suggesting Final Topics: Based on the nodes weights, we rank and report the final topics. (E.g. the node with highest weight is reported as the first-rank topic, etc.)

We employ the above algorithm to suggest topics for a benchmark data set (explained next) using our ontology and a baseline ontology. In this way, we can compare the two ontologies, by evaluating the results generated by the above algorithm for the benchmark data set using each ontology.

Baseline Taxonomy: To generate the baseline ontology, we start by 684 animal names provided by Kozareva et al. [15]. We refer to this list as the *initial animals list*. Next, we retrieve all the hypernyms located on the paths connecting these titles to the *animal* node in WordNet. This Ontology/Taxonomy (called *WNTax*) contains respectively 2,687 and 5,624 concepts and [*type_of*] links. We should add that, one can not simply start with the *animal* node in WordNet and find all its descendants/hyponyms to create a domain specific taxonomy. This is due to the fact that many terms have several meanings and each may have different hyponyms. Exhaustively following such hyponyms starting from a general term (e.g *animal*) would result in a very general taxonomy.

OntoHarvester Taxonomy: Using the animal ontology generated by OntoHarvester earlier, and considering only *type_of* links, we create a taxonomy referred to as *OHTax*.

Benchmark Data Set: We create a simple benchmark by collecting the abstract of 440 animal names in the *initial animals list* from Wikipedia. Unfortunately, the other items in this list do not have any abstract in Wikipedia. We also assign three sets of possible topics to each abstract in the benchmark. i) *Title*: the animals name, ii) *Direct Cats.*: the direct categories provided for the *Title* by Wikipedia, and iii) *Indirect Cats.*: the indirect or secondary categories that are the ancestor of the largest number of direct categories for the abstracts. These possible topics are compared with the suggested topics by TID to evaluate the quality of taxonomies.

TABLE V
RESULTS OF TOPIC IDENTIFICATION USING FOR TAXONOMIES.

Taxonomy Name	Topic Source	Rank-1 Topic # (%)	Avg. Rank
OHTax	Title	**133** (30.2%)	**2.77**
OHTax	Direct Cats.	**78** (17.7%)	**4.47**
OHTax	Indrct. Cats.	**128** (29.1%)	**6.68**
WNTax	Title	56 (12.7%)	5.36
WNTax	Direct Cats.	35 (7.9%)	6.09
WNTax	Indrct. Cats.	97 (22.0%)	6.88

We feed *WNTax* and *OHTax* to the TID algorithm to suggest topics for abstracts in the benchmark. That is for each abstract, we run TID using the two taxonomies and get the ordered list of suggested topics. The result of this experiments are provided in Table V. As shown in the third column of the table (in bold font), when *OHTax* is used, TID is able to find the exact title for 30.2% of the abstracts in the benchmark with its first ranked suggested topics. This is more than twice the same value for *WNTax*. Similar results hold for the other two topic sources (*Direct Cats.* and *Indirect Cats.*). In all cases, *OHTax* significantly outperforms *WNTax*, due to having more concepts related to the domain of animals.

Considering the top-20 topics suggested by TID, we compute the number of correctly identified topics from each topic source. Then for each abstract, we determine the position (rank) of the correct topic in ordered list. The average ranks is shown in the fourth column of Table V. Again, TID's results for *OHTax* are of higher quality than those for *WNTax*. More specifically, the average rank of the correctly reported topics using *OHTax* for the *Title* case is 2.77, which is about half of the same value for *WNTax*.

The above experiment mainly indicates that although WordNet (and similarly other general ontologies) may be of high quality, they are not comprehensive enough for domain-specific applications such as the one mentioned in this subsection. OntoHarvester, on the other hand, is able to use very small seed ontology (or existing ontologies) and a small set of application-specific textual documents to learn more comprehensive ontologies for the purpose of the application. This makes OntoHarvester a very powerful and practical tool for learning domain-specific ontologies.

V. RELATED WORK

In the last two decades, many highly supervised approaches have been proposed to generate ontologies [6][33][20][7]. Due to the need for human validation, most of these approaches are too expensive to scale up. Thus, many works employ statistical techniques or machine learning techniques to automatically extract ontologies [19][26][29][32][17][27]. However, these

approaches typically suffer from issues such as requiring very large training sets, needing to be re-trained for new domains, highly rely on structured or semi-structured data sets, and not being able to learn from small text corpora.

To address these issues, more NLP-based techniques have been proposed in recent years. Lin and Pantel [18] automatically find (binary) relation structure between terms and use them to generate ontology. Many similar works have tried to extract similar relations through shallow NLP approaches and use various techniques to refine them and create the final Ontology [3][28][15][25][16]. One of the few works showing that using *deeper* NLP-based techniques, such as parse tree mining, can provide more accurate results is CRCTOL proposed by Jiang and Tan in [14]. In CRCTOL, parse trees are used to find candidate terms (from NPs for instance), and other statistical techniques are employed to extract the concepts from the candidate terms. Unlike OntoHarvester, most of these works either (i) use very limited and shallow NLP-based techniques, (ii) use a limited and fixed number of patterns (mostly borrowed from Hearst [11]) to mine the text or parse trees, or (iii) separate the concept extraction phase from relation extraction. Thus, they do not gain much over the pure statistical techniques.

VI. Conclusion

We introduced OntoHarvester, an unsupervised domain-specific ontology generator from free text using TextGraphs—rich structures describing grammatical relations between terms in the text. By utilizing patterns in such graphs, our system recognizes ontological relations between the existing concepts and candidate terms, and populates the initial ontology in an iterative fashion. This enables us to derive high-quality ontologies using a modest corpus and a small seed. In fact, the ontologies so produced outperform (in terms of precision and recall) those produced by previous techniques, including high-quality NLP-based systems such as CRCTOL. Even more significant is the fact that using the ontologies produced by OntoHarvester, the performance of the intended application will improve significantly as demonstrated by the topic-identification application discussed in the paper.

VII. Acknowledgments

We would like to sincerely thank Alan Koenig, R. Augustus Rick, Adam Quaal, and Sanghyun Cho for their invaluable help on grading the results of our system.

References

[1] Sparql query language for rdf. http://www.w3.org/TR/rdf-sparql-query/, 2008.

[2] The stanford parser: A statistical parser. http://nlp.stanford.edu/software/lex-parser.shtml, 2013.

[3] M. Banko, M. J. Cafarella, S. Soderl, M. Broadhead, and O. Etzioni. Open information extraction from the web. In *IJCAI*, 2007.

[4] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. Dbpedia - a crystallization point for the web of data. *J. Web Sem.*, 7(3):154–165, 2009.

[5] K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD Conference*, pages 1247–1250, 2008.

[6] D. Bourigault. Surface grammatical analysis for the extraction of terminological noun phrases. In *COLING*, pages 977–981, 1992.

[7] P. Drouin. Term extraction using non-technical corpora as a point of leverage. *TERMINOLOGY*, 9:99–116, 2003.

[8] E. Drymonas, K. Zervanou, and E. G. M. Petrakis. Unsupervised ontology acquisition from plain texts: The *ontogain* system. In *NLDB*, pages 277–287, 2010.

[9] G. Furnas, T. Landauer, L. Gomez, and S. Dumais. The vocabulary problem in human-system communication. *Commun. ACM*, 30(11):964–971, Nov. 1987.

[10] T. R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 6:199–220, 1993.

[11] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *COLING*, pages 539–545, 1992.

[12] J. Hoffart, F. M. Suchanek, K. Berberich, E. Lewis-Kelham, G. de Melo, and G. Weikum. Yago2: exploring and querying world knowledge in time, space, context, and many languages. In *WWW*, 2011.

[13] M. Janik and K. Kochut. Training-less Ontology-based Text Categorization. In *Workshop on ESAIR*, Mar. 2008.

[14] X. Jiang and A.-H. Tan. Crctol: A semantic-based domain ontology learning system. *JASIST*, 61(1):150–168, 2010.

[15] Z. Kozareva and E. H. Hovy. A semi-supervised method to learn and construct taxonomies using the web. In *EMNLP*, 2010.

[16] J. Krishnamurthy and T. M. Mitchell. Which noun phrases denote which concepts? In *Proceedings of ACL: HLT*, pages 570–580, PA, USA, 2011.

[17] C.-S. Lee, Y.-F. Kao, Y.-H. Kuo, and M.-H. Wang. Automated ontology construction for unstructured text documents. *Data Knowl. Eng.*, 60(3):547–566, Mar. 2007.

[18] D. Lin and P. Pantel. Dirt @sbt@discovery of inference rules from text. In *KDD*, pages 323–328, 2001.

[19] S. Loh, L. K. Wives, and J. P. M. de Oliveira. Concept-based knowledge discovery in texts extracted from the web. *SIGKDD Explor. Newsl.*, 2(1):29–39, June 2000.

[20] A. Maedche and S. Staab. Semi-automatic engineering of ontologies from text. In *SEKE*, Chicago, IL, 2000.

[21] H. Mousavi, S. Gao, and C. Zaniolo. Ibminer: A text mining tool for constructing and populating infobox databases and knowledge bases. *PVLDB*, 6(12):1330–1333, 2013.

[22] H. Mousavi, D. Kerr, and M. Iseli. A new framework for textual information mining over parse trees. In *ICSC*, pages 185–188, 2011.

[23] H. Mousavi, D. Kerr, M. Iseli, and C. Zaniolo. Ontoharvester: An unsupervised ontology generator from free text. In *CSD TR #130003, UCLA*, 2013.

[24] H. Mousavi, D. Kerr, M. Iseli, and C. Zaniolo. Mining semantic structures from syntactic structures in free text documents. In *CSD TR #140005, UCLA*, 2014.

[25] R. Navigli, P. Velardi, and S. Faralli. A graph-based algorithm for inducing lexical taxonomies from scratch. In *IJCAI*, 2011.

[26] P. Pantel and D. Lin. A statistical corpus-based term extractor. In *Canadian Conference on AI*, pages 36–46, 2001.

[27] A. G. Parameswaran, H. Garcia-Molina, and A. Rajaraman. Towards the web of concepts: Extracting concepts from large datasets. *PVLDB*, 3(1):566–577, 2010.

[28] H. Poon and P. Domingos. Unsupervised ontology induction from text. In *ACL*, pages 296–305, 2010.

[29] T. Quan, S. Hui, A. Fong, and T. Cao. Automatic generation of ontology for scholarly semantic web. *ISWC*, pages 726–740, 2004.

[30] R. Snow. Semantic taxonomy induction from heterogenous evidence. In *In Proceedings of COLING/ACL 2006*, pages 801–808, 2006.

[31] V. Stoyanov and C. Cardie. Topic identification for fine-grained opinion analysis. In *COLING*, pages 817–824, Stroudsburg, PA, USA, 2008.

[32] Q. T. Tho, S. C. Hui, A. C. M. Fong, and T. H. Cao. Automatic fuzzy ontology generation for semantic web. *IEEE Trans. on Knowl. and Data Eng.*, 18(6):842–856, June 2006.

[33] A. Voutilainen. Nptool, a detector of english noun phrases. *CoRR*, cmp-lg/9502010, 1995.

[34] W. Wong, W. Liu, and M. Bennamoun. Ontology learning from text: A look back and into the future. *ACM Comput. Surv.*, 44(4):20, 2012.

[35] F. Wu and D. S. Weld. Automatically refining the wikipedia infobox ontology. In *WWW*, pages 635–644, 2008.

[36] W. Wu, H. Li, H. Wang, and K. Q. Zhu. Probase: a probabilistic taxonomy for text understanding. SIGMOD, pages 481–492, 2012.