Ontology Learning and its Application in Software-Intensive Projects

Jin Guo
Advised by: Dr. Jane-Cleland Huang
School of Computing
DePaul Univeristy
jguo9@cdm.depaul.edu

ABSTRACT

Software artifacts, such as requirements, design, source code, documentation, and safety-related artifacts are typically expressed using domain-specific terminology. Automated tools which attempt to analyze software artifacts in order to perform tasks such as trace retrieval and maintenance, domain analysis, program comprehension, or to service natural language queries, need to understand the vocabulary and concepts of the domain in order to achieve acceptable levels of accuracy. Domain concepts can be captured and stored as an ontology. Unfortunately, constructing ontologies is extremely time-consuming and has proven hard to automate. This dissertation proposes a novel approach for semi-automated ontology building that leverages user-defined trace links to identify candidate domain facts. It uses a variety of web-mining, Natural Language Processing, and machine learning techniques to filter and rank the candidate facts, and to assist the user in building a domain-specific ontology. The benefits of the constructed ontology are described and evaluated within the context of automated trace link creation.

CCS Concepts

•Software and its engineering → Software creation and management; *Traceability;* **•Computing methodologies →** *Ontology engineering;*

Keywords

Ontology Learning; Software Domain Knowledge

1. INTRODUCTION

Many Software Engineering tasks depend upon the careful analysis of textually-rich software artifacts. For example, coverage analysis requires a business analyst or engineer to review natural language requirements, understand their intent, and evaluate whether they are sufficiently satisfied in the design or implemented system. Trace link creation requires an analyst to examine the semantics of terms that exist in a pair of artifacts, such as a software requirement and a source code class, in order to determine if the artifacts are conceptually related. Unfortunately, in practice, there is often a significant term mismatch between different artifacts even when they are conceptually related. Human domain experts intrinsically know and understand the concepts that connect dissimilar terms and are able to bridge this gap effectively; however, such information is typically not available to automated techniques. In practice, most algorithms performing text-related tasks rely primarily on the syntax of the text or semantics which can be inferred dynamically at runtime. As a result, automated approaches often fail to establish relations correctly. This explains why trace retrieval solutions deliver far from perfect recall and precision – often retrieving fewer than 90% of the targeted links, sometimes at extremely low accuracy [7].

The concepts needed to compensate for term mismatches, as well as the means of reasoning over related concepts, can be captured in the form of an Ontology. Thomas R. Gruber defined ontology as " an explicit specification of a conceptualization" [4]. It provides a formal definition of objects, concepts, properties, and their inter-relationships. It also supports logical operations which allow us to infer relationships between concepts. Ontologies can be either general or domain-specific. The most popular general ontology is WordNet, a lexical ontology in which nouns, verbs, adjectives and adverbs are organized as interlinked synsets [10]. Other general ontologies have been generated from Wikipedia or derived and integrated from multiple knowledge sources [13, 14]. However, these types of general ontology do not contain the kinds of technical terminology typically used to engineer software intensive systems. Domain specific ontologies are normally created manually by domain experts, often using data mining techniques to extract candidate terms from domain documents. Examples include MeSH[1] of medical thesaurus , SNOMED CT[2] of clinical health terminology and the GeneOntology[3] of the roles of genes and gene products in organisms. Each of these has been created at significant cost and human-effort.

The goals of this research are two-fold. First, to deliver a practical approach for constructing domain-specific ontologies. To this end we propose a novel approach for semi-

ICSE '16 Companion, May 14 - 22, 2016, Austin, TX, USA

ISBN 978-1-4503-4205-6/16/05…$15.00

DOI: http://dx.doi.org/10.1145/2889160.2889264

[1]https://www.nlm.nih.gov/mesh/
[2]http://www.ihtsdo.org/snomed-ct
[3]http://geneontology.org/

automated ontology building that leverages user-defined traceability information to identify candidate domain facts. It uses a variety of web-mining, Natural Language Processing (NLP), and machine learning techniques to filter and rank the candidate facts, and to assist the user in building a domain-specific ontology. The second goal is to develop techniques that utilize the ontology to support specific software engineering tasks. Our focus is primarily on trace link generation, however we also explore other areas. Paradoxically, our approach leverages the traceability information available in one or more software projects, to discover domain concepts and to construct an ontology which then improves our ability to accurately generate trace links and to analyze textual information in future projects. The following research questions will be addressed through this work:

RQ1: How can ontology be learned for a specific software domain? To what extent does the generated ontology capture accurate and useful domain facts?

RQ2: How can the constructed ontology be leveraged to improve the accuracy of text-intensive software engineering tasks such as trace-retrieval or natural language Q&A?

RQ3: Can the ontology learned from a subset of projects be effectively used to improve software engineering tasks in other projects from the same domain?

The remainder of this paper is laid out as follows. In Section 2, we first introduce our current progress toward answering those research questions. In Section 3, we propose a plan for addressing the topics and problems raised during our research. Related work in the area of Ontology Construction and its Application in Software Engineering domain is introduced in Section 4. We finally state the expected contribution of our work in Section 5.

2. CURRENT PROGRESS

Software-intensive projects are specified and modeled using domain terminology. The domain concepts, and their interrelationships, can be documented in an ontology and used to enhance communication, clarify domain assumptions, enable reuse, and facilitate meaningful analysis of text-based artifacts. Our current work includes an initial solution for leveraging traceability data to build ontology, and then incorporating that data into the trace-retrieval task. Further, we have performed an initial investigation which suggests that ontology can be used to improve the tracing task across other projects in the domain.

2.1 Ontology Learning

Figure 1 depicts a trace link between a regulation for preserving privacy of personal data in a medical information system and a system-level requirement. It shows the domain facts which are needed to establish traceability between the two artifacts. A domain fact contains an element from the source artifact (e.g., "access right"), an element from the target artifact (e.g., "privileges"), and a relation type (e.g., "is-a"). Our goal is to use the trace links to generate *pairings* of source-target elements and then leverage web-mining, NLP, and machine learning techniques to search for evidence that the pairing potentially represents a meaningful domain fact. The number of element pairings grows very large as the length of source and target artifacts increase, and the majority of pairings do not represent valid domain facts. Our challenge is therefore to filter out unrelated ones and only present the most likely pairings as candidate facts to the user for further inspection.

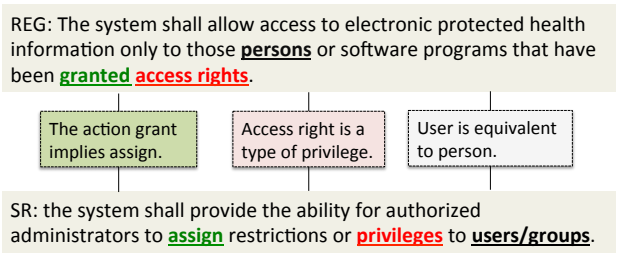

Figure 1: The Trace Link Leverages Domain Facts

The first step is to identify *pairings* of source and target elements. Based on our analysis of the types of facts contained in a diverse set of ontologies, we focus attention on nouns, noun phrases and verbs and extract them using a Part-of-Speech tagger and a phrase extractor [5]. Each candidate fact contains one term or phrase extracted from the source artifact and one from the target artifact.

We then gather evidence from different knowledge resources that include *Lexico-Syntactic Pattern (LSP) Matching*, *Association Rule Mining*, *Topic Modeling*, and *Semantic Relatedness*. *LSPs* represent recurrent expressions in natural language text [8]. *LSP Matching* searches domain documents and project artifacts with the aim of finding direct evidence to support a candidate fact. For example, discovering a phrase such as *user privileges including access rights* provides direct evidence that "access right" is a type of "privilege". *Association Rule Mining* calculates how often combinations of terms/phrase pairs appear across all existing trace links. *Topic Modeling* computes the extent to which terms and phrases relate to the same topics. *Semantic Relatedness* describes the semantic similarity of the terms/phrases based on external ontology source. We use WordNet[10] in our experiment. Each technique provides a different degree of support for a domain fact.

We use the evidence (or lack of evidence) provided by each knowledge source to train a Random Forest classifier. Training is based upon a manually created training set. In our initial evaluation, the Domain Fact classifier achieved F-Measure score ranging from 0.68 to 0.78 across three datasets from the transportation, medical device and health information domains. Candidate facts are then validated by the user before being added to the domain ontology. We show the overview of our system in Figure 2.

2.2 Ontology Application

Our purpose for building ontology is to support a variety of software engineering tasks - including traceability, ambiguity analysis, and design satisfaction assessment. Here, we illustrate the potential usefulness of a domain ontology for generating and explaining software trace links.

In [5, 6], we propose *DoCIT*, a domain-specific intelligent traceability solution for the trace retrieval task. DoCIT first uses NLP techniques to match terms/phrases onto concepts in the ontology. For each identified verb, DoCIT then uses a set of syntactic mapping rules to construct intermediate representations called *Action Frames*. Action Frames contain several thematic roles which represent the most important content related to each verb. For instance, *Agent* stands for

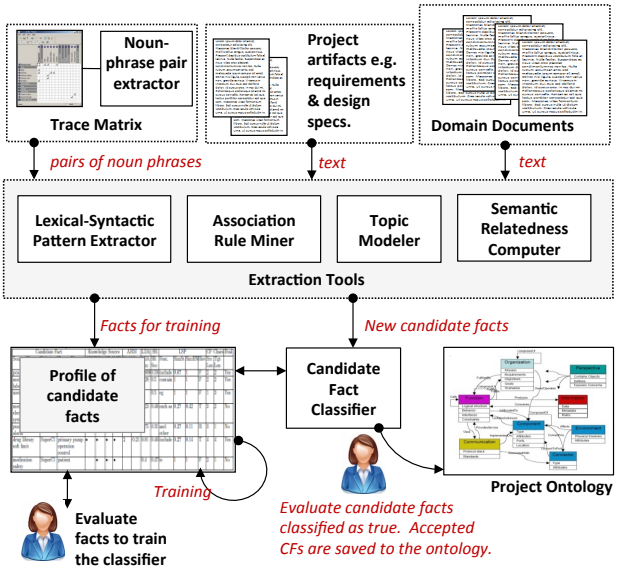

Figure 2: Ontology Learning System Overview

Table 1: Two DoCIT Action Frames are Linked via a Heuristic matching Process

A1: The OBM shall support reception and decomposition of Wayside Status Messages (WSM).	
Action	reception
Semantic Group	Receptive
Recipient	obm
Theme	wayside status message
A2: The Wayside Segment shall transmit information to the Automobile Segment in the form of Wayside Status Messages.	
Action	transmit
Semantic Group	Transmissive
Agent	wayside segment
Recipient	automobile segment
Theme	wayside status message

a doer of the action and *Theme* represents the object upon which the action is performed. Other identified thematic roles include *Recipient*, *Location*, and *Initial Location*. Finally DoCIT uses link heuristics to establish trace links by matching action frames across source and target artifacts. Link heuristics are defined in terms of a set of matching rules defining exactly how thematic roles across the pair of action frames must be matched according to the ontology. For instance, the heuristic rule applied to the two action frames shown in Table 1 requires their corresponding *Recipients* to exhibit an exact, hierarchical, or compositional match, and our ontology contains the fact that ***OBM is located on Automobile segment*** which complies with this requirement. This, and other ontology facts, supports validation of this trace link.

Based on an ontology built for the domain of Driver-Optional Highway System, DoCIT significantly improved the accuracy of the generated trace links, achieving over 90% recall with over 80%precision. These results are much higher than those obtained using term-based approaches [6].

2.3 Ontology Reuse

One of the main reasons for developing an ontology is to enable knowledge reuse. If the ontology learned from one project can be applied to others in the same domain, it would effectively solve the "cold-start" problem for many software engineering tasks when new projects are just launched.

We therefore collected ten datasets from the Patient Healthcare domain including open source products, IT healthcare standards, requirements exemplars, and feature descriptions for commercial products. They all trace to the USA′s Health Insurance Portability and Accountability Act (HIPAA)[4] regulations. We constructed a domain ontology using trace links between HIPAA and the largest dataset, named CCHIT[5]. We used this ontology to support the trace retrieval task for the remaining nine datasets using an Ontology-Enhanced VSM model. In this model, the query is augmented with domain ontology entities that are related to the query′s original terms/phrases with the weight of their semantic relatedness.

Our initial experiment showed that the Ontology-Enhanced model outperformed VSM across all nine datasets. Furthermore, in five of the nine datasets, it achieved a significant improvement in Mean Average Precision of more than 30%.

3. PROPOSED WORK

The work completed so far serves as a proof-of-concept that Ontology can be used to effectively improve software engineering activities such as Trace-Retrieval. Planned work will include the following:

3.1 Ontology Learning

Our proof-of-concept approach to ontology learning incorporates several different techniques and then combines the evidence they provide through use of a domain-fact classifier. Planned work includes improving and evaluating each of the individual techniques through activities such as (1) specifying more LSP expressions, (2) improving the phrase-extractor, (3) exploring other sources of evidence, (4) evaluating the efficacy of each individual technique, (5) investigating techniques for semantically typing relationships, and (5) investigating alternate ways to merge evidence into a final result. We will comparatively evaluate our approach against existing techniques for ontology generation [11]. The ontology will be evaluated both qualitatively using human assessors and quantitatively by using it to generate trace links and evaluating their accuracy. This work is expected to be completed by September 2016.

3.2 Link Heuristic Searching

As previously explained an ontology not only contains concepts and relationships, but also provides the means for reasoning over those relationships. The DoCIT approach introduced in Section 2.2 performs this task using a set of heuristics which are defined for pairs of semantic groups in terms of conceptual matches between thematic roles. Given an ontology, we will use search-based techniques to automate the learning of these heuristics. This approach is only feasible in the presence of large datasets containing tens of thousands of trace links. We plan to utilize datasets from

[4]Health Insurance Portability and Accountability Act of 1996, 42 U.S.C.A. 1320d to d-8 (West Supp. 1998)

[5]The Certification Commission for Healthcare Technology http://www.cchit.org

the transportation, and insurance domains provided by our existing industrial collaborators. Evaluation will be conducted against other datasets from transportation, healthcare, and space domains including the ones used in [5, 6], and additional ones from our industrial collaborators and the traceability research community.

4. RELATED WORK

There is a significant body of work in the area of ontology construction. Techniques can largely be classified as linguistic, statistical, and machine learning. Initial efforts toward ontology building focused on matching lexico-syntactic patterns that occur repeatedly in free text [8]. However, the recall achieved using these methods is normally low due to limitations in defining sufficient patterns. Iterative bootstrapping techniques were introduced to overcome this problem. But bootstrapping suffers from low precision caused by semantic drift, which occurs when the extracted relations inevitably drift from the meaning of the original seeds [1]. To mitigate this issue, strategies have been applied to constrain the learning process, such as type-checking relation arguments [16]. Researchers have also applied probabilistic models to evaluate and rank the extracted relations [17].

More recently, researchers have focused efforts on mining ontology facts from large-scale public-facing knowledge bases, including structured or at least semi-structured data, in an automated fashion [13, 14]. Unfortunately, for many software engineering domains, very limited such structured information is currently available. Ontology mining therefore requires complex reasoning to extract and interpret important domain phrases and concepts. Mining structured knowledge is therefore not directly applicable to the kinds of problems we are targeting in this paper.

Software Engineering researchers have proposed using ontology to support software engineering tasks. However, the focus is often on creating ontology to describe requirements [9], services [15], UML diagrams [12], etc. Few software engineering researchers have tackled the problem of ontology creation. A major exception is the work by Gacitua et al. who proposed a framework for ontology construction [3] and explored different methods for assembling knowledge [2]. In contrast to their work we leverage traceability data to guide and constrain the ontology discovery process.

5. EXPECTED CONTRIBUTION

This research aims to construct domain-specific ontologies leveraging traceability data and utilizing web-mining, NLP and Machine learning techniques. We plan to deliver algorithms and tools for building ontology and enhanced trace-retrieval algorithms that leverage the generated ontology.

6. ACKNOWLEDGEMENTS

The work in this paper was partially funded by the US National Science Foundation Grant CCF:1319680.

7. REFERENCES

[1] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, and T. M. Mitchell. Toward an architecture for never-ending language learning. In *In AAAI*, 2010.

[2] R. Gacitua and P. Sawyer. Ensemble methods for ontology learning - an empirical experiment to evaluate combinations of concept acquisition techniques. In *7th IEEE/ACIS Conf. on Computer and Information Science, ICIS 2008, USA*, pages 328–333, 2008.

[3] R. Gacitua, P. Sawyer, and P. Rayson. A flexible framework to experiment with ontology learning techniques. *Knowl.-Based Syst.*, 21(3):192–199, 2008.

[4] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

[5] J. Guo, J. Cleland-Huang, and B. Berenbach. Foundations for an expert system in domain-specific traceability. In *Requirements Engineering Conference (RE), 2013 21st IEEE International*, pages 42–51. IEEE, 2013.

[6] J. Guo, N. Monaikul, C. Plepel, and J. Cleland-Huang. Towards an intelligent domain-specific traceability solution. In *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering*, pages 755–766. ACM, 2014.

[7] J. H. Hayes, A. Dekhtyar, and S. K. Sundaram. Advancing candidate link generation for requirements tracing: The study of methods. *IEEE Trans. Softw. Eng.*, 32(1):4–19, 2006.

[8] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics, 1992.

[9] I. Jureta, J. Mylopoulos, and S. Faulkner. A core ontology for requirements. *Applied Ontology*, 4(3-4):169–244, 2009.

[10] G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[11] D. Movshovitz-Attias and W. W. Cohen. Grounded discovery of coordinate term relationships between software entities, 2015.

[12] K. Robles, A. Fraga, J. Morato, and J. Lloréns. Towards an ontology-based retrieval of UML class diagrams. *Information & Software Technology*, 54(1):72–86, 2012.

[13] M. Ruiz-Casado, E. Alfonseca, and P. Castells. Automatising the learning of lexical patterns: An application to the enrichment of wordnet by extracting semantic relationships from wikipedia. *Data Knowl. Eng.*, 61(3):484–499, 2007.

[14] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *Proc. 16th international conference on World Wide Web*, pages 697–706. ACM, 2007.

[15] B. Verlaine, Y. Dubois, I. Jureta, and S. Faulkner. Towards conceptual foundations for service-oriented requirements engineering: bridging requirements and services ontologies. *IET Software*, 6(2):85–102, 2012.

[16] C. Wang, A. Kalyanpur, J. Fan, B. K. Boguraev, and D. Gondek. Relation extraction and scoring in deepqa. *IBM Journal of Research and Development*, 56(3.4):9–1, 2012.

[17] W. Wu et al. Probase: A probabilistic taxonomy for text understanding. In *2012 ACM SIGMOD Int'l Conf on Management of Data*, SIGMOD '12, pages 481–492, New York, NY, USA, 2012. ACM.