Semi-automated Ontology Creation for High-level Service Classification

Knarig Arabshian, Peter Danielsen

Alcatel-Lucent, Bell Labs
Murray Hill, NJ, USA
knarig.arabshian@alcatel-lucent.com
peter.danielsen@alcatel-lucent.com

Abstract— **In this paper, we propose a LexOnt, a semi-automatic ontology creation tool for high-level service ontology. LexOnt uses the Programmable Web directory of services as its corpus of data along with Internet sources such as Wikipedia in order to generate common terms within a service domain and determine generic properties for classification. Currently, Programmable Web classifies services in a flat categorization where each service is manually classified within a single category. Search is limited to attributes that are not related to the semantics of the service, such as protocol or messaging type. When using an ontology for the service API descriptions, services can be automatically classified and queried for according to their attributes. Additionally, services can then be distributed in an ontology-based service discovery system such as GloServ so that semantic registration and querying of services becomes possible.** **Keywords: semi-automated ontology creation, OWL, service discovery, service composition**

I. Introduction

In recent years, the availability of services on the World Wide Web has surged. In order to make good use of these services, both human and software consumers require knowledge about existing services. Thus, the need for automatic service discovery and composition is critical. A current technological trend is in creating web service mashups. The concept of a mashup is to allow users to create their own content from different types of sources such as websites, RSS Feeds [1], or Flickr [2]. A user is able to filter tailored information on a personal page to view and share with others. It is not necessary for a user to know how to create websites, but can do so simply by bringing different components together via a simplified user interface. For example, the Google Maps API [3] is often used in conjunction with location-based web services. Currently, there are a number of web services in various domains, such as social media or mapping services that offer their APIs to be used in mashup applications. Programmable Web [4] is one such directory that offers a listing of Web service APIs.

Creating mashups of web services is a difficult process since these services exist as separate entities on the web. Every site has its databases modeled in a specific way, causing semantically equivalent properties to be defined differently, since data is not easily shared across different domains in the Internet.. As a result, service discovery and composition across different service domains becomes a tedious process. With the use of Semantic Web technologies, such as description logic ontologies and reasoners, data linking becomes a seamless effort. Thus, ontologies are now being used to describe and classify services. Currently, there are a number of ontology languages that are under development for representing a service. A few are: Web Ontology Language for Services (OWL-S) [5], Web Services Modeling Ontology (WSMO) [6], and Semantic Web Services Language (SWSL) [7]. All of these use some form of the Web Ontology Language (OWL) [8] which is the de facto ontology standard within the W3C and used by a few of the standard description logic reasoners such as Pellet [9] and RacerPro [10].

One of the limitations in using ontologies for services is in creating a generic description of a service domain. By a generic description, we mean a high-level description of a service that is not just a taxonomic classification, but also describes general properties shared across the service domain. This means finding common properties that exist within a service domain, such that all services in that domain have at least those property assignments. For example, all social networking services have at least user profile and friend list properties as well as some kind of sharing attribute; or all location-based services have at least location as a property.

Currently, most of the semi-automated ontology generation tools either have an established taxonomy or a well-defined corpus, such as in the fields of biology and medicine which aid domain experts in creating ontologies. There are tools which analyze generic text to semi-automatically create an ontology, but these ontologies are taxonomic hierarchies and do not indicate property descriptions of the classes. As we have indicated above, there is a need for ontology descriptions of different service domains to ease the discovery and composition of services. In order to make the use of ontologies for web service descriptions prevalent, a tool is required that will allow anyone to create a high-level ontology description of a service domain given a corpus of data.

In this paper, we propose a LexOnt, a work-in-progress for a semi-automatic ontology creation tool for a high-level service ontology. LexOnt uses the Programmable Web directory of services as its corpus of data along with Internet sources such as Wikipedia [11] in order to generate common terms within a service domain and determine generic properties for classification. Currently, Programmable Web classifies services in a flat categorization where each service is manually classified within a single category. Search is limited to attributes that are not related to the semantics of the service, such as protocol or messaging type. When using an ontology for the service API descriptions, services can be automatically classified and queried for according to their attributes. Additionally, services can then be distributed in an ontology-based service discovery system such as GloServ [12] so that semantic registration and querying of services becomes possible.

The main contribution of this paper is LexOnt's novel algorithm and implementation for semi-automatic ontology creation. LexOnt combines text analysis for term generation on the PW Service API data and Wikipedia sources, using the Lucene pacakage [13], with ontology term matching in order to semi-automatically generate an ontology description of the PW Service class. We describe the details of the work-in-progress algorithm and implementation in this paper.

The paper is divided as follows: Section II describes related work in semi-automatic ontology creation; Section III describes the algorithm and implementation of LexOnt; and we conclude in Section IV.

II. Related Work

Most of the related work closest to our work involves semi-automated ontology creation for taxonomic hierarchies or domains that already have some kind of structural description. Machine learning techniques are used on text corpora alongside an already existing ontological description of the domain to generate an ontology for that specific dataset. There is work that uses clustering for semi-automatic construction of ontologies from parsed text corpora [14], [15], creating taxonomic hierarchies [16] or topic ontologies [17]. The work closest to ours are those that involve finding property relationships between concepts. A few systems have been proposed: TextToOnto [18], Ontolearn [19], OntoLT [20].

The main difference betwee LexOnt and these systems is that they start off with some kind of ontological or structured description in addition to the text corpora whereas we start with only the descriptions of the service API in order to determine the relationships. Thus, we also use other types of data to describe the domain of the service such as matching common terms from Wikipedia to the common terms within the service API to generate important terms and phrases. Also, in our work, we are concentrating mainly on describing service domains. Although our techniques may be applied to all types of domains, it works best for describing high-level service domains as the properties that are shared within a service class are feasible to deduce with semi-automatic means since they are limited to a few generic properties.

III. Semi-automated Ontology Creation using LexOnt

LexOnt is a semi-automatic ontology creation tool. Although it can be applied to all types of textual data, we have geared it toward the domain of services since it is possible to describe and classify a service on a high-level, like in a yellowpage directory, where common properties are shared across all services within that domain. We distinguish services by characteristic features of that service.

As mentioned above, an example would be the social networking service. All social networking services have user profiles, friend lists and some kind of sharing attribute. There are many social networking services available on the Internet today and these are distinguished by the required feature of a user profile and a friend list and then certain types of sharing features. For example, Facebook's sharing features would include: photos, videos, status updates, or applications, whereas Twitter's sharing feature would be status updates. However, there are many other types of social networks that share information not necessarily captured by the most commonly used SNs. For example, travel information, live video chats, book reviews, or blogs. When searching for a social networking service that allows one to share a specific feature, these property values need to be defined. Thus, the goal of LexOnt is to enable a user to distinguish common terms within a category such that these terms can be applied as distinguishing features of a service domain. Once these terms are chosen, they are added as "hasFeature" properties to the ontology.

It is unrealistic to assume that an ontology can be created in a purely automated fashion. Ontologies are often subjective descriptions of a given domain which require human evaluation. However, we can semi-automate this process by analyzing a corpus of data within a domain and aiding a user who may not be completely familiar with a domain to choose terms that may describe this domain, and then automating the instantiation of these attributes. Furthermore, as the ontology is being created for a domain, the information within it can also be used to give higher weights for the terms within the corpus that match the terms that have been instantiated in the ontology.

We are currently researching various algorithms to determine the best way to generate a top-n list of terms. Unlike other semi-automatic ontology generators that focus on a single domain, we are targeting all types of service domains and users who may not be experts within the domain. The novelty in LexOnt's algorithm is twofold: 1) use information from HTML text describing service APIs, wikipedia articles describing that service domain and a thesaurus ontology for synonymous terms as our corpus for generating a top-n list of words and phrases 2) semi-automate the construction of the ontology by labeling terms that have been assigned manually to the ontology with higher weight within the corpus and then creating or asserting property assignments within the ontology for subsequent iterations. Thus, we incorporate the current state of the ontology into the corpus of data itself to assign higher weights to those terms which are already assigned in the ontology and then regenerate a list of terms from the text data given the current ontological terms. Text analysis for generating a top-n list is done using Lucene. We are still looking at various text analysis and machine learning algorithms within the Lucene package to see which is best to use for the top-n term or phrase generation.

A. Algorithm

1) Semi-automatic ontology creation process: The pseudocode below describes the process of ontology creation. This is an interactive process where the system starts with a set of service classes, generates a top-n list given the initial corpus of data, then the user chooses relevant terms, enters it into the system as property-value assignments, the system then adds this information onto the ontology and then uses the newly consturcted ontology as part of its corpus by regenerating a new list of top-n terms. The user can maually add information to the ontology using an ontology editor such as Protege [21].

LexOnt:

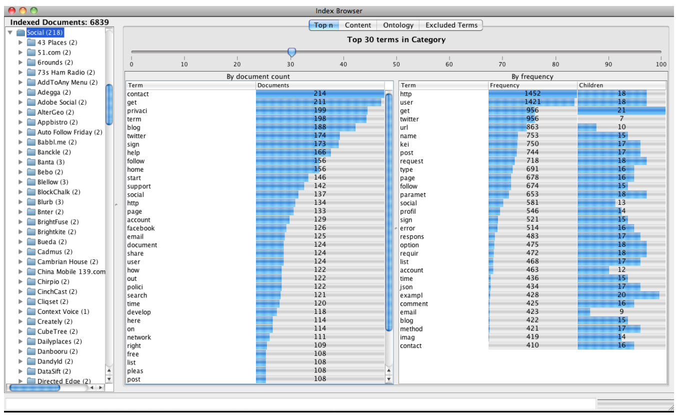
- Creates an OWL ontology of service classes.
- Generates a top-n list of terms using: PW service instance data, wikipedia and thesaurus ontology.
- Displays this in a GUI format seen in Figure 1

User:

- For a given category, look through the top-n terms or phrases generated for each PW instance.
- Choose keywords to assign as features.
- Enter these terms into the "Ontology Processing" tab in LexOnt to run ontology processing code

LexOnt:

- For every PW instance that has these terms in the top-n list, create a corresponding OWL instance and add a hasFeature property assigned to that term.
- Use the newly constructed ontology to regenerate a new list of top-n terms by assigning these newly created terms within the ontology with higher weights.



Indexed Documents: 6839

- Social (218)
  - 43 Places (2)
  - 51.com (2)
  - 6rounds (2)
  - 73s Ham Radio (2)
  - AddToAny Menu (2)
  - Adegga (2)
  - Adobe Social (2)
  - AlterGeo (2)
  - Appbistro (2)
  - Auto Follow Friday (2)
  - Babbl.me (2)
  - Banckle (2)
  - Banta (3)
  - Bebo (2)
  - Blellow (3)
  - BlockChalk (2)
  - Blurb (3)
  - Bnter (2)
  - BrightFuse (2)
  - Brightkite (2)
  - Bueda (2)
  - Cadmus (2)
  - Cambrian House (2)
  - China Mobile 139.com
  - Chirpio (2)
  - CinchCast (2)
  - Cliqset (2)
  - Context Voice (1)
  - Creately (2)
  - CubeTree (2)
  - Dailyplaces (2)
  - Danbooru (2)
  - Dandyld (2)
  - DataSift (2)
  - Directed Edge (2)

Top 30 terms in Category

By document count

Term	Documents
contact	214
get	211
privaci	199
term	198
blog	188
twitter	174
sign	173
help	166
follow	156
home	156
start	146
support	142
social	137
http	134
page	133
account	129
facebook	126
email	125
document	124
share	124
user	124
how	122
out	122
polici	122
search	121
time	120
develop	118
here	114
on	114
network	111
right	109
free	108
list	108
pleas	108
post	108

By frequency

Term	Frequency	Children
http	1452	18
user	1421	18
get	956	21
twitter	956	7
url	863	10
name	753	15
kei	750	17
post	744	17
request	718	18
type	691	16
page	678	16
follow	674	15
paramet	653	18
social	581	13
profil	546	14
sign	521	15
error	514	16
respons	483	17
option	475	18
requir	472	18
list	468	17
account	463	12
time	436	15
json	434	17
exampl	428	20
comment	425	16
email	423	9
blog	422	15
method	421	17
imag	419	14
contact	410	16

Fig. 1. LexOnt Semi-automatic Ontology Generation Tool

B. Implementation

The corpus consists of web pages from each of the APIs listed in Programmable Web. We first fetch the Programmable Web listing for each API. A listing contains links to web pages maintained by the API owner. Those pages are fetched, stripped of their HTML, and added to a Lucene index. The index maintains the content of each page, and its Programmable Web category and API. We use the Lucene index to find Programmable Web categories, to find APIs within a category, to query for terms and their frequencies, and to build top-n lists of term frequencies within a category and within an API.

C. Discussion

From our initial use of LexOnt, we are seeing promising results. Initially we processed a category that we are familiar with, Social Network. In this category, it was quickly obvious that the features of the Social Network class comprise of a user profile, friend list and then a set of things that are being shared. After looking through a few instances, it became clear that the top-n list included terms about what was being shared. A quick look at the stripped HTML data confirmed this as well. We then tried processing a category that we were not familiar with. We chose the Advertising category. Once again, after looking through a few instances, we saw that the top-n terms included words that were related to the types of advertising offered such as search engine marketing, banners, email marketing, etc. As these terms were chosen, more instances that matched these keywords were also found and automatically created and assigned those properties. Thus, with just a basic top-n keyword generating algorithm of frequence count, we have already seen promising results. We hope that with the incorporation of wikipedia, the thesaurus ontology and generated ontology data, these terms and phrases will become even more accurate.

IV. Future Work and Conclusion

For future work, we are looking at the best way to generate top-n list of terms. Currently we look at frequency count. The Lucene package offers a number of text analysis tools which we will work through to see what works best. Additionally, for the thesaurus ontology, we are going to use a generic thesaurus API and build an ontology of terms as the ontology is generated.

Eventually, we would like to describe services not just on a high-level but in its functional details. Allowing standard descriptions of a service's inputs, outputs, pre and post conditions will aid in autmomatic service composition. Once we have the high-level discovery phase automated with this ontology, we plan on continuing this work to see how we can generate an ontology description of the service which include its functional details.

In conclusion, we have presented our work-in-progress, LexOnt, a semi-automatmic ontology generator that aids in the ontology creation of a high-level service ontology. It uses the Programmable Web directory of services, Wikipedia, thesaurus ontology and the current state of the generated ontology to suggest relevant terms that may be incorporated within the ontology. The LexOnt builds the ontology iteratively, by interacting with the user, taking in terms that the user has chosen, adding these to the ontology and then regenerating list of terms. From our initial findings, we have determined that LexOnt is a useful tool to generate a high-level ontology description of a domain, specifically for users who are not domain experts.

References

[1] Rss: Really simple syndication specifications. [Online]. Available: http://www.rssboard.org/rss-specification

[2] Flickr. [Online]. Available: http://www.flickr.com

[3] Google maps api. [Online]. Available: http://code.google.com/apis/maps/index.html

[4] Programmable web. [Online]. Available: http://www.programmableweb.com

[5] Owl-s (semantic markup for web services). [Online]. Available: http://www.w3.org/Submission/OWL-S/

[6] Wsmo: Web services modeling ontology. [Online]. Available: http://www.wsmo.org/

[7] Semantic web services language. [Online]. Available: http://www.daml.org/services/swsl/

[8] Owl: Web ontology language. [Online]. Available: http://www.w3.org/2004/OWL/

[9] Pellet. [Online]. Available: http://clarkparsia.com/pellet/

[10] Racer system. [Online]. Available: http://www.racer-systems.com/

[11] Wikipedia. [Online]. Available: http://www.wikipedia.org

[12] K. Arabshian and H. Schulzrinne, "An ontology-based hierarchical peer-to-peer global service discovery system," *Journal of Ubiquitous Computing and Intelligence (JUCI)*, vol. 2, pp. 133ï¿$\frac{1}{2}$–144, December.

[13] Apache lucene. [Online]. Available: http://lucene.apache.org/java/docs/index.html

[14] G. Bisson, C. Nedellec, and L. Canamero, "Designing clustering methods for ontology building: The mok workbench," in *Ontology Learning Workshop, The 14th European Conference on Artificial Inteligence (ECAI)*, Berlin, Germany, 2000.

[15] M. Reinberger and P. Spyns, "Discovering knowledge in texts for the learning of dogma-inspired ontologies," in *In Proceedings of the Ontology Learning and Population Workshop, The 16th European Conference on Artificial Inteligence (ECAI)*, Valenci, Spain, 2004.

[16] L. S.-T. S. S. P. Cimiano, A. Pivk, "Learningtaxonomicrelations from heterogeneous evidence." in *Ontology Learning and Population Workshop The 16th European Conference on Artificial Inteligence (ECAI)*, Valenci, Spain, 2004.

[17] D. M. B. Fortuna, M. Grobelnik, "Ontogen: Semi-automatic ontology editor," in *Human Interface, Part II, HCII 2007*, 2007.

[18] A. Maedche and S. S. Staab, "Semi-automatic engineering of ontologies from text." in *12th International Conference on Software Engineering and Knowledge Engineering*, 2000.

[19] R. Navigli and A. G. P. Velardi, "Ontology learning and its application to automated terminology translation," in *IEEE Intelligent Systems, vol. 18:1*, 2003.

[20] S. Buitelaar and D. Olejnik, "A protege plug-in for ontology extraction from text based on linguistic analysis," 2004.

[21] Protege ontology editor and knowledge acquisition system. [Online]. Available: http://protege.stanford.edu/