Metric-Based Ontology Learning

Hui Yang
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, PA, 15213
huiyang@cs.cmu.edu

Jamie Callan
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, PA, 15213
callan@cs.cmu.edu

ABSTRACT

Ontology learning is an important task in Artificial Intelligence, Semantic Web and Text Mining. This paper presents a novel framework for, and solutions to, three practical problems in ontology learning. An incremental clustering approach is used to solve the problem of unknown group names. Learned models at each level of an ontology address the problem of no control over concept abstractness. A metric learning module moves beyond the limitation of traditional use of features and incorporates heterogeneous semantic evidence into the learning process. The metric-based learning framework integrates these separate components into a single, unified solution. An extensive evaluation with WordNet and Open Directory Project data demonstrates that the method is more effective than a state-of-the-art baseline algorithm.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

Ontology Learning, Concept Abstractness, Ontology Metric

1. INTRODUCTION

Ontology learning is an important task in Artificial Intelligence, Semantic Web and Text Mining. Given a set of concepts extracted from a text corpus, organizing them into the correct hierarchy for knowledge representation is the problem of ontology learning. The learned concept ontology is not only a hierarchical summary for a text corpus, but also a tool for further reasoning and inferencing by other data analysis tasks, for instances, gene retrieval based on gene ontology and word sense disambiguation based on a noun hyponym ontology. A well-defined ontology can greatly help research in other related fields. WordNet [7] and Open Directory Project (ODP) [1] are two such excellent examples, which are widely used in Information Retrieval and Natural Language Processing.

Despite manually-constructed ontologies, such as WordNet and ODP, there are numerous ontologies learned by automatic and semi-automatic algorithms [1, 3, 4, 5, 8, 9, 10, 12, 13, 14, 16]. Many algorithms use hierarchical clustering, in particular bottom-up hierarchical clustering, as the main framework. It is due to the fact that ontologies are hierarchies in nature. The traditional ontology construction process is as follows: firstly detect concept candidates, then cluster them bottom-up based on the agglomerative clustering algorithm: merge similar concepts together, find a name for a new group and move to the next higher level to repeat the process, finally form a tree-structured ontology. Note that in the above process, there is a difficult step - "find a name for a new group". Given a group of concepts, automatically finding a term or a phrase to best describe the group members, is still an unsolved problem. One strategy to tackle (or bypass) this problem is to replace the bottom-up approach by an incremental one. In this paper we present an incremental clustering framework to build an ontology step by step by considering each concept candidate in turn and putting them into the correct positions in the hierarchy. In our framework, all concepts in the final ontology come from an available concept candidate set. Thus the problem of unknown group names is avoided in our approach.

Another major shortfall of previous ontology learning algorithms has been the ignorance to concept abstractness. Current technologies treat concepts at different abstraction levels of an ontology in the same way, however, it is clear that abstract concepts, such as "science", "economy", "thought" and concrete concepts, such as "hopo camp", "mercury pollution", "polar bear" are different in many ways. Concrete concepts are considered to be physical entities with characteristic shapes, parts, materials, etc., whereas abstract concepts lack physical attributes [6]. Concrete concepts are likely to lie at the lower levels of an ontology, whereas abstract concepts lie at the higher levels since they cover broader knowledge and subsume more descendent concepts than concrete concepts. Moreover, simple string pattern matching works well for detecting hyponyms for concrete concepts, for instance, "hopo camp" is a hyponym for "camp", which follows the simple rule - "NP1 NP2 is a hyponym for NP2", whereas such pattern matching seldom works for abstract concepts, for instance, "chemistry" is a hyponym for "science" but does not follow the above rule. The algorithm we present in this paper is designed to address the differ-

[1] http://www.dmoz.org/

CIKM'08, October 26–30, 2008, Napa Valley, California, USA.

ences from different concepts categories and model them in the learning framework to produce more sensible results.

A third contribution of our approach is the flexibility to incorporate heterogenous semantic evidence. Previous technologies either use only one type of semantic evidence to infer all relationships in an ontology (for instance, substring matching in the myGrid system [12], document statistics in the subsumption approach [13]) or use one type of semantic evidence for a particular subtask in ontology learning (for instance, lexico-syntactic patterns for conditional probability prediction [14], cosine document similarity for measuring concept differences [3]). By ontology metric learning, our work provides a general framework which allows flexible inclusion of all kinds of semantic evidence into the learning process, and selects the best suitable features depending on the actual needs.

Our approach presents solutions to the above three problems simultaneously: (1) by taking the incremental clustering approach, we solve the problem of unknown group names; (2) by learning statistical models for concepts at different abstraction levels, we explicitly address concept abstractness; (3) by a separate metric learning module, we incorporate heterogenous syntactic and semantic features into the learning process. The metric-based learning framework transforms the ontology learning task into an optimization problem and offers a unified solution to these three problems.

The following of this paper is organized as: Section 2 gives our definition and assumptions about ontology space. Section 3 details the metric-based ontology learning framework. Section 4 describes the application and implementation of our framework for noun hyponym ontology construction. Section 5 evaluates the approach with WordNet and ODP data and compares it with a state-of-the-art algorithm. Section 6 concludes the paper.

2. THE ONTOLOGY SPACE

To have a theoretical formulation of the learning framework, we give a formal definition of an ontology and related concepts in this section. After that, we give the three important assumptions about ontology space which will be applied into our algorithm design and implementation.

2.1 Definition of Ontology Space

Definition 1. An **ontology** is a data model `T` that represents a set of concepts $\{c_1, c_2, ..., c_n\}$ within a domain `D` and a set of relationships `R` between those concepts.

$$\texttt{T=(C,R|D)}$$

where

$$\texttt{C} = \{c_1, c_2, ..., c_{|\texttt{C}|}\}$$

$$\texttt{R} = \{(c_1, c_2), (c_1, c_3), ...(c_{|\texttt{C}|-1}, c_{|\texttt{C}|})\}.$$

Imagine that all the human knowledge has been put into a space, which we call the knowledge space. Each ontology is a part of this knowledge space and focuses on one domain. There is a many-to-one mapping from an ontology to a domain in the knowledge space. To develop our assumptions about an ontology, let us introduce a new concept - ontology space.

Definition 2. An **ontology space** `S(T)` is the space occupied by an ontology `T` in the entire knowledge space. It is represented by the size of the space, which is measured by the amount of information included in the ontology.

$$\texttt{S(T)=Info(T)}$$

where `Info(.)` is a function to measure the amount of information.

Definition 3. A **full ontology space** $\texttt{S(T:T} \in full)$ is the ontology space occupied by an ontology `T` which roots from a concept c whose descendent concepts all present.

Definition 4. A **partial ontology space** $\texttt{S(T:T} \notin full)$ is the ontology space occupied by an ontology `T` which roots from a concept c some of whose descendent concepts are missing.

2.2 Assumptions of Ontology Space

Given the definition of ontology space, we make the following assumptions about ontology space and its properties.

ASSUMPTION 1. *Constant Total Assumption. The amount of information in a full ontology space is a constant.*

$$\texttt{Info(T:T} \in full) = K$$

where K is a constant.

ASSUMPTION 2. *Minimum Evolution Assumption. During the process of completing an full ontology space, there are many partial ontology spaces. The ontology $\texttt{T}^{n+1}$ associated with the next ontology space $\texttt{S}^{n+1}$ from the current ontology space $\texttt{S}^n$ is the one introduces the least changes in the information between these two spaces:*

$$T^{n+1} = \arg\min_{T'} \Delta Info(T^n, T')$$

where

$$\Delta Info(T^n, T') = \|Info(T^n) - Info(T')\|.$$

ASSUMPTION 3. *Abstractness Assumption. Concepts at the same abstraction level share the same characteristic function as their `Info(.)` function. Concepts at different abstraction level have different characteristic functions.*

$\forall$*abstraction level i:*

$$\forall c \in i, Info(T_c) = f_i(T_c)$$

$\forall$*abstraction level i, j:*

$$\forall i, j, i \neq j, f_i(.) \neq f_j(.)$$

where T_c is the ontology roots from concept c. $f_i(.)$ is the characteristic function for concepts at level i.

3. A METRIC-BASED LEARNING FRAMEWORK

In the knowledge space, ontologies and domains follow a many-to-one mapping. For instance, for a biological domain, there are functional ontology for the functional relationships among the actions, and nominal ontology for hyponym relationships between noun concepts [11, 12]. Given so many kinds of ontologies, it is time-consuming to tailor algorithms for a particular kind of ontology. In this section, we present

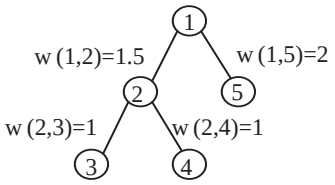

Figure 1: Illustration of Ontology Metric

a ontology learning framework which is general enough for many different ontology learning tasks.

In section 3.1, we introduce an important concept - ontology metric, for this metric-based learning framework. In section 3.2, we give the formal problem formulation for the ontology learning task. In section 3.3, we present a greedy optimization algorithm to construct the ontology based on the minimum evolution assumption.

3.1 Ontology Metric

A metric is a distance measure, $d(.,.)$, between all data pairs (i, j) in a set S. Formally, it is a function $d : S \times S \longrightarrow \mathrm{R}_+$. As a valid matric, d has to fulfill several criteria:

- Non-negativity. $d(i, j) \geq 0$
- Symmetricity. $d(i, j) = d(j, i)$
- Equality Condition. $d(i, j) = 0 \Leftrightarrow i = j$
- Triangular Inequality. $d(i, j) + d(j, s) \geq d(i, s)$

An ontology metric is a metric which functions on an ontology $\mathtt{T}$. Formally, it is a function $d : C \times C \longrightarrow \mathrm{R}_+$, where C is the set of concepts in $\mathtt{T}$. Note that the graph structure of an ontology is actually a tree, a graph without cycles. In graph theory [15], a tree metric is defined as a function on the set of leaf nodes in the tree. Similar to but different from the tree metric, ontology metric is specially designed for the metric-based ontology learning framework and is a function on the entire set of nodes.

The ontology metric $d_{(T,w)}$ on an ontology $\mathtt{T}$ with edge weights w for any data pair (i, j) in the concept set $\mathtt{C}$ is defined as the sum of all edge weights along the path between the data pair:

$$d_{(T,w)}(i, j) = \sum_{e_{ij} \in P(i,j)} w(e_{ij})$$

where $P(i, j)$ is the set of edges defining the path from concept i to j.

Figure 1 gives an illustration of ontology metric on a 5-node ontology. Since ontology metric is still a metric, it has all four properties for a metric function. In particular, given the definition of ontology metric, the last property - triangular inequality, falls into a special situation, where the sum of the ontology metrics for the ending points on two connected paths equals the ontology metric for the ending points on the entire path:

$$d(i, j) + d(j, s) = d(i, s).$$

Moreover, note that when two data points directly connect, the path between them reduces to the edge between them. Therefore, in this case, the ontology metric is the edge weight. Given that a metric should be non-negative, all of the edge weights should also be non-negative. This property can be ensured by the four point condition: for any four concepts (i, j, s, t) in $\mathtt{T}$,

$$d(i, j) + d(s, t) \leq \max\left(d(i, s) + d(j, t), d(i, t) + d(j, s)\right).$$

3.2 Problem Formulation

We define the problem of ontology learning as the construction of a full ontology $\hat{\mathrm{T}}$ given a set of concepts $\mathtt{C}$ and an initial partial ontology $\mathrm{T}^0(\mathrm{S}^0, \mathrm{R}^0)$, where $\mathrm{S}^0 \subseteq \mathtt{C}$. Note that the initial ontology could be empty. The process of ontology learning then starts from the initial partial ontology T^0 and keeps adding the concepts in set $\mathtt{C}$ one by one into T^0, until the full ontology is formed (all of the concepts in $\mathtt{C}$ are added).

At each concept insertion step, a new partial ontology $\mathtt{T}$ is formed. We define the amount of information carrying in an ontology to be the sum of all ontology metrics in $\mathtt{T}$:

$$Info(T) = \sum_{i<j, c_i, c_j \in C} d(c_i, c_j) \tag{1}$$

By the abstractness assumption of ontology space in section 2, concepts at different abstraction levels result in different characteristics. To capture these differences, we define the amount of information for an abstraction level L as:

$$Info(L) = \sum_{i<j, c_i, c_j \in L} d(c_i, c_j) \tag{2}$$

where L is a subset of concepts lying at the same level of ontology $\mathtt{T}$. For example, in Figure 1, node 1 belongs to level L_1, and node 2 and 5 belong to level L_2.

By the minimum evolution assumption of ontology space, the ontology $\mathtt{T}^{n+1}$ associated with the next ontology space $\mathtt{S}^{n+1}$ from the current ontology space $\mathtt{S}^n$ is the one introduces the least changes in the information between these two spaces. Therefore, within our framework, we define the goal of ontology learning is to find the optimal full ontology $\hat{\mathrm{T}}$ such that the information changes since the initial partial ontology T^0 are the least, i.e., to find:

$$\hat{T} = \arg\min_{T'} \Delta Info(T', T^0) \tag{3}$$

where T' is a full ontology, i.e., the set of concepts in T' is $\mathtt{C}$.

3.3 The Optimization Algorithm

We have formulated the ontology learning task into an optimization problem. In this section, we present the optimization algorithm and show that the learning framework offers a unified solution to the three problems we have mentioned earlier in section 1.

To find the optimal solution for equation 3, we need to find the optimal set of concepts and the optimal set of relationships for T'. Since the optimal set of concepts for a full ontology is always $\mathtt{C}$, the only unknown part is the optimal set of pairwise relationships $\hat{\mathrm{R}}$. Thus, the optimization formula in equation 3 can be formulated into an equivalent problem, i.e., to find:

$$\hat{R} = \arg\min_{R'} \Delta Info(T(C, R'), T^0(S^0, R^0)) \tag{4}$$

Based on the optimization solution of $\hat{\mathrm{R}}$, every concept in $\mathtt{C}$ is placed in the optimal position in an ontology. Note that $\mathtt{C}$ is a given input, which contains all concepts that will

appear in the ontology. Since there is no need to find new concepts that are not in C, the first problem of unknown group names in the bottom-up approaches is resolved trivially in our framework.

Note that in the framework, concepts are added incrementally into the ontology. At each concept insertion step, by the minimum evolution assumption, the resulting ontology after a new concept is inserted should be the one gives the least information change. Therefore, the updating function for the resulting set of relationships $R^{(n+1)}$ after a new concept z is inserted can be calculated as:

$$R^{(n+1)} = \arg\min_{R'} \Delta Info(T(S^n \cup \{z\}, R'), T(S^n, R^n)) \quad (5)$$

By plugging in the definition of the information change function $\Delta Info(.,.)$ in section 2 and the formulas in equation 1, we have the updating function as:

$$R^{(n+1)} = \arg\min_{R'} \| \sum_{c_x, c_y \in S^n \cup \{z\}} d(c_x, c_y) - \sum_{c_x, c_y \in S^n} d(c_x, c_y) \| \quad (6)$$

The above updating function can be transformed into a minimization problem, which is defined as:

$$\min u$$

$$\begin{aligned} \text{subject to} \quad & u \leq \textstyle\sum_{c_x, c_y \in S^n \cup \{z\}} d(c_x, c_y) - \sum_{c_x, c_y \in S^n} d(c_x, c_y) \\ & u \leq \textstyle\sum_{c_x, c_y \in S^n} d(c_x, c_y) - \sum_{c_x, c_y \in S^n \cup \{z\}} d(c_x, c_y) \\ & x < y \end{aligned}$$

The above minimization problem is defined following the minimum evolution assumption, hence we call it the "minimum evolution" objective function.

Further, to address the second problem of no control over concept abstractness, we model concept abstractness explicitly into another objective function. Based on the abstractness assumption (see section 2), each level L_i is characterized by its own function f_i. We define the characteristic functions as a linear interpolation of some underlying feature functions H_i. The optimization problem is the least square fit of the amount of information at level L_i, and the linear interpolation score $W_i^T H_i$. Therefore, the "abstractness" optimization objective function can be defined as:

$$\min \sum_i \| Info(L_i) - W_i^T H_i \|^2$$

By plugging in the definition of the amount of information for an abstraction level and the underlying feature functions, the "abstractness" objective function becomes:

$$\min \sum_i \| \sum_{c_x, c_y \in L_i} d(c_x, c_y) - \sum_j w_{i,j} h_{i,j}(c_x, c_y)) \|^2$$

where $w_{i,j}$ is the weight for the j^{th} underlying feature function at level L_i, $h_{i,j}(.,.)$ is the j^{th} underlying feature function for concept pairs at level L_i.

Note that modelling of concept abstractness is done by approximating characteristic functions for each abstraction level as a linear interpolation of a set of underlying feature functions. In theory, there is no specific constraint on quantity and definitions of the underlying feature functions. In practice, the selection of good feature functions may depend on a specific application. Nevertheless, the design of the learning framework offers a flexible inclusion of heterogenous features and hence tackle the third problem of limited use of features. In section 4, we will demonstrate more details of our use of heterogenous features in an application.

In this metric-based learning framework, both "minimum evolution" and "abstractness" objectives need to be satisfied . To optimize multiple criteria, it needs to reach the Pareto optimality [2]. Since the two optimization problems are both convex, the Pareto optimal can be obtained by scalarization. That is to say, introduce a variable λ to control the contribution of each objective within the range of 0 to 1. The multiple criterion optimization function becomes:

$$\min \lambda u + (1 - \lambda) v$$

$$\begin{aligned} \text{subject to} \quad & u \leq \textstyle\sum_{c_x, c_y \in S^n \cup \{z\}} d(c_x, c_y) - \sum_{c_x, c_y \in S^n} d(c_x, c_y) \\ & u \leq \textstyle\sum_{c_x, c_y \in S^n} d(c_x, c_y) - \sum_{c_x, c_y \in S^n \cup \{z\}} d(c_x, c_y) \\ & v = \textstyle\sum_i \| \sum_{c_x, c_y \in L_i} d(c_x, c_y) - \sum_j w_{i,j} h_{i,j}(c_x, c_y)) \|^2 \\ & x < y \\ & 0 \leq \lambda \leq 1 \end{aligned}$$

The multiple criterion optimization leads to the following greedy optimization algorithm, which at each concept insertion step, produces a new partial ontology by adding to the old one a new concept z, and a new set of relationships $R_{(z,.)}$, which minimizes the multi-criterion objective function. The algorithm is shown as the following:

foreach $z \in C \setminus S$
  $S \leftarrow S \cup \{z\}$;
  $R \leftarrow R \cup \{\arg\min_{R_{(z,.)}} (\lambda u + (1 - \lambda) v)\}$;
Output $T(S, R)$

The above optimization algorithm present a general incremental clustering procedure to construct ontologies. By minimizing the ontology structure changes and modelling concept abstractness at each step, it finds the optimal position of each concept in the ontology hierarchy. In the next section, we will show an application of this framework on a common ontology construction task.

4. NOUN HYPONYM ONTOLOGY CONSTRUCTION

Among all kinds of ontologies, perhaps noun hyponym ontology is most commonly used. Although our proposed learning framework can be used to construct any ontology, in this section, we focus on the most common task - noun hyponym ontology construction.

WordNet and ODP are two available human-built ontologies. We extract subdirectories from WordNet and ODP, and test our algorithm to investigate its ability to reconstruct these subdirectory hierarchies. Each subdirectory becomes one dataset in our experiments, and each dataset is an ontology. In WordNet subdirectory extraction, we only use the word sense occurring in a particular subdirectory. Therefore there is no multiple senses for a word in one ontology. In ODP subdirectory extraction, we parse the topic lines, such as "Topic r:id="Top/Arts/Movies"", in the XML dataset to get concepts, such as "arts" and "movies" in this example, along the paths. In total, there are 100 subdirectories[2] , 50 each extracted from WordNet and ODP.

[2]The 50 WordNet subdirectories are from 12 topics: gathering, professional, people, building, place, milk, meal, water, beverage, alcohol, dish and herb. The 50 ODP subdirecto-

With the extracted ontologies, we create both training and testing datasets. The training data is in the same format of an extracted ontology: a set of concepts and a set of pairwise relationships between the concepts. The testing data only contain the set of concepts in the corresponding ontology (since we need to find the relationships). Note that in other tasks such as constructing ontology for a given corpus, concept candidates need to be extracted by noun phrase mining and named entities recognition [16].

In the following subsections, we first show how to estimate the ontology metric for a new pair of concepts and then we address the use of heterogenous features again by relating our solution (for the third problem of limited use of features) to the task of noun hyponym ontology construction.

4.1 Estimating Ontology Metric

It is crucial to learn a good ontology metric for each pair of concepts in an ontology in this metric-based learning framework. In order to apply the multi-criterion optimization algorithm in section 3, it needs to know the ontology metric for the testing data. The idea is to learn the metric from the training data.

In the training data, the ontology metric $d(x, y)$ for concept pair (x,y) is generated by assuming every edge weight as 1 and summing up all the edge weights along the path from concept x to y. We assume that there are some underlying feature functions which measure the semantic distance from concept x to concept y. A weighted combination of these functions will approximate the ontology metric for (x,y):

$$d(x, y) = \sum_j w_j h_j(x, y)$$

where w_j is the j^{th} weight factor for $h_j(.,.)$, the j^{th} feature function.

In our experiments, the estimation and prediction of ontology metric are done by ridge regression.

4.2 Heterogenous Features

As part of the proposed general framework, feature functions are used more than once. In both ontology metric estimation (section 4.1) and abstractness optimization (section 3), pairwise underlying feature functions are exploited. In theory, the design of the learning framework allows us to embed as many feature functions as we want into the framework as long as they are indicators of semantic difference/similarity between two concepts. In the application of noun hyponym ontology construction, we focus on the following features:

Google KL-Divergence The top 1000 Google snippets are downloaded using each concept as a query issued to Google search engine. The snippets are built into a language model for each concept. This feature function then measures the KullbackLeibler divergence (KL divergence) between the language models associated with any two concepts.

Wikipedia KL-Divergence The entire Wikipedia corpus is downloaded and indexed by Indri[3]. The top 100 relevant documents returned by Indri after being queried by each concept are built into a language model. This feature function then measures the KL divergence between the language models associated with any two concepts.

Google Minipar Syntactic Distance The top 1000 Google documents are downloaded for every pair of concepts after querying Google search engine by a concatenation of a concept pair (quotation marks are added to group terms in a phrase). The documents are then split into sentences and each sentence is parsed by Minipar[4]. This feature function returns the edge distance between two concepts in a syntactic parse tree.

Google Lexico-Syntactic Patterns The top 1000 Google documents are downloaded for every pair of concepts after querying Google search engine by a concatenation of a concept pair. We mine the top 10 frequent lexico-syntactic patterns in the above Google collection for a seed hypernym-hyponym pair. We obtain patterns such as "NP_A is a NP_B", "NP_A, a NP_B", etc, where NP_A refers to the hypernym, NP_B refers to the hyponym in the pair. Those patterns are then used to score difference between an unknown pair of concepts. This feature function returns a vector of distance scores for any two concepts.

Term Co-occurrence This feature function returns the total document count (appears on the first page of Google search results) in log scale after querying Google search engine by a concatenation of a concept pair.

Word Length Difference This feature function simply returns the difference of number of words in two concepts.

These features vary from simple statistics to complicated syntactic patterns, basic word length feature to comprehensive Web-based features, which is heterogenous. The flexible design of our learning framework allows us to use all of them and even allow different feature sets for concepts at different abstraction levels. We study the interactions between features and different abstraction levels in section 5.

Table 1: Statistics of Datasets

Statistic	WordNet	ODP
total #datasets	50	50
total #concepts	1964	2210
average #concepts	39	44
max #concepts	86	104
average depth	5.5	5.9
max depth	9	8
level w/ most concepts	4	4

5. EVALUATION

To evaluate the proposed framework for ontology learning, we apply the techniques developed for noun hyponym ontology construction to reconstruct subdirectories from WordNet and ODP. The 100 datasets (as introduced in section 4) containing more than 4,000 concepts, are from various topics. Table 1 shows statistics about the ontology structure of the data.

In section 5.1 we describe our evaluation methodology, followed by three sets of experiments in the following sections. The experiments are designed based on the three foci of this paper. Section 5.2 tests on the performance of the main ontology learning task for noun hyponym ontology

ries are from 16 topics: computers, robotics, intranet, mobile computing, database, operating system, linux, tex, software, computer science, data communication, algorithms, data formats, security multimedia and artificial intelligence.

[3]http://www.lemurproject.org/indri

[4]http://www.cs.ualberta.ca/ lindek/minipar.htm

construction and also compares our system with a strong baseline system. Section 5.3 evaluates the impact of concept abstractness on system performance. Section 5.4 studies the interaction of individual feature and abstraction level, and their joint impact on system performance.

5.1 Methodology

We evaluate the quality of the learned ontologies by comparing them with the gold standards - original ontologies obtained from WordNet and ODP. A hypernym-hyponym pair is the basic evaluation unit. From both a test run and the gold standard, a list of hypernym-hyponym pairs are generated and compared.

Precision, recall and F1 measures are the evaluation metrics used in the following sets of experiments. Since ontology is in a tree-structure and concepts at the higher levels are harder to obtain due to their higher abstractness, we weigh concepts at different levels with different weights so that abstract concepts count more. If the levels in an ontology are indexed in ascending order from top-down (level 1 indicates the top level), the number of correctly returned hypernym-hyponym pairs can be calculated as:

$$\sum_{(i,j)} \mathbf{1}(\text{pair } (i,j) \text{ is returned correctly}) * \frac{max\_level - level(j) + 1}{\sum_{l=1}^{max\_level} l}$$

where $\mathbf{1}(.)$ is the indicator function, $level(.)$ is the function returns the level index of a concept, max_level is the maximum depth of an ontology. The number of returned concepts and the number of concepts in the gold standard can be calculated in a similar way by changing the definition of the indicator function.

Leave-one-out cross validation is used to measure the learning effect across different training and test datasets. For the 50 datasets from either WordNet or ODP, we pick 49 of them to generate training data, and test on 1 dataset. We repeat the process for 50 times, with different training and test sets at each time, and report the averaged precision, recall and F1 across all 50 runs. The results reported in section 5.3 and 5.4 are obtained exactly in this way. In section 5.2, we perform leave-one-out cross validation on a different total of datasets by gradually increasing the number of training datasets while still testing on one dataset. The averaged precision, recall and F1 across that total number of datasets, ranging from 2 to 50, are reported.

5.2 Performance of Ontology Learning

To test the performance for the main ontology learning task, experiments in this section compare our metric-based framework with a state-of-the-art baseline system [14] from Stanford University. The baseline system is a strong baseline, which takes a probabilistic approach for ontology construction. At each concept insertion step, they maximize the joint probability of the whole hierarchy, which is different from our work. Another difference is that in their original work, only lexico-syntactic patterns are used to predict conditional probability scores, whereas in our work, we can easily incorporate various kinds of semantic features. To have a fair comparison, we re-implement their framework to add the same set of features as in our work.

Figure 2 and 3 demonstrate the overall system performance of our system and the baseline system for WordNet and ODP datasets. For WordNet datasets, precision of the baseline system ranges from 0.7 to 0.84 (which is consistent with what have been reported in [14]), whereas precision of the proposed algorithm ranges from 0.78 to 0.93, with an absolution gain of 8%-9%. For ODP datasets, precision of the baseline system ranges from 0.68 to 0.87, whereas precision of the proposed algorithm ranges from 0.79 to 0.95, with an absolute gain of 8%-11%. In summary, the proposed metric-based approach outperforms the baseline by an absolute gain around 10% for both WordNet and ODP.

From Figure 2, we can see that with the total number of concepts increasing, the system performance are improving. However, Figure 3 shows that the improvement of precision, recall and F1, stops after adding a certain number of concepts as the training data, and then becomes stable after adding more. This may be attributed to the observation that there is more noise in ODP than in WordNet. For example, under "artificial intelligence", there are "neural networks", "natural language" and "academic departments", where "academic departments" should not be a hyponym here.

5.3 Impact of Concept Abstractness

In this section, we study the impact of modelling concept abstractness on system performance. Figure 4 and 5 show the changes of system performance when varying the coefficient λ in the Pareto objective (see section 3.3). Hence λ is a coefficient to adjust the contributions of minimum evolution objective and abstractness objective. As $\lambda \to 0$, the system relies more on the abstractness objective whereas as $\lambda \to 1$, the system relies more on the minimum evolution objective. λ is an indicator to see the impact of modelling concept abstractness in this work.

When $\lambda = 1$, the system purely relies on the minimum evolution objective. As λ decreases, the contribution of concept abstractness increases, and the system performance improves till reaching the maximum, where λ lies in the range of 0.7 to 0.8. After reaching the maximum, as λ keeps decreasing, the contribution of concept abstractness increases, but the system performance drops. The optimal λ values, 0.8 for WordNet and 0.7 for ODP, are used in experiments in section 5.2 and 5.4.

It shows that a good combination of both objectives is important. Modelling concept abstractness indeed improves the overall performance as comparing to not modelling it at all (when $\lambda = 1$). However, the major contributor is still the metric-based optimization framework, i.e., the minimum evolution objective.

5.4 Impact of Various Features

In this section, we study the effect of using heterogenous features and the joint impact of features and concept abstractness on system performance. Figure 6 and 7 demonstrate the impact of various semantic features on different levels of concepts in an ontology. On the x-axis are indices of the levels, each of which is used in turn to model concept abstractness. Level 1 is the top level. Level=0 means there is no abstraction level modelled. On the y-axis are different features used in this work, each of which is switched on in turn. On the z-axis are F1 values of leave-one-out cross validation averaged over 50 datasets. The surface indicates the F1 values, whereas the shading indicates different features.

Both figures show that the system performance boosts when abstract levels (levels 2-3), opposite to concrete levels (level 4-8), are modelled. This can be attributed to the ob-

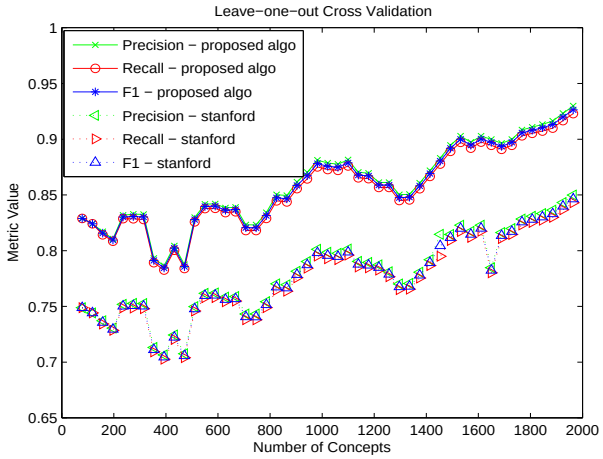

Figure 2: Precision, Recall and F1 for WordNet

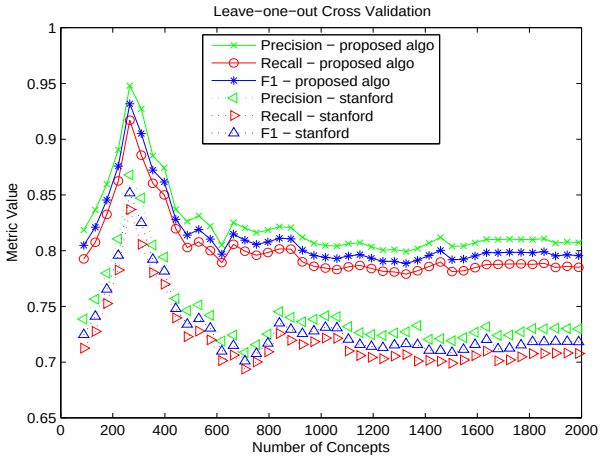

Figure 3: Precision, Recall and F1 for ODP

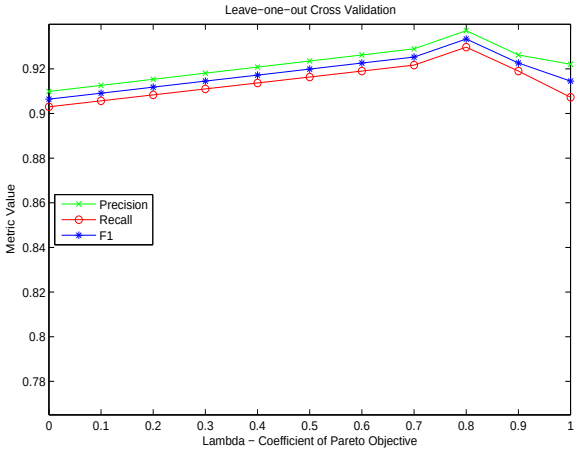

Figure 4: Impact of Concept Abstractness for WordNet

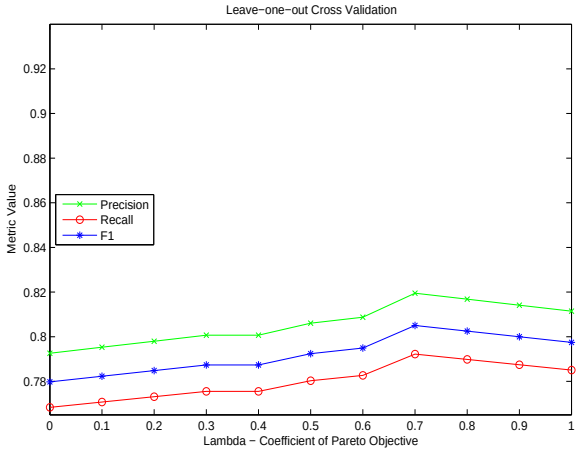

Figure 5: Impact of Concept Abstractness for ODP

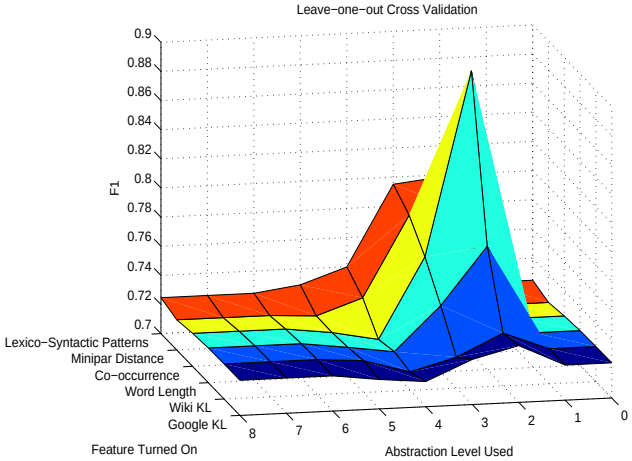

Figure 6: Feature Impact on Different Abstraction Level for WordNet

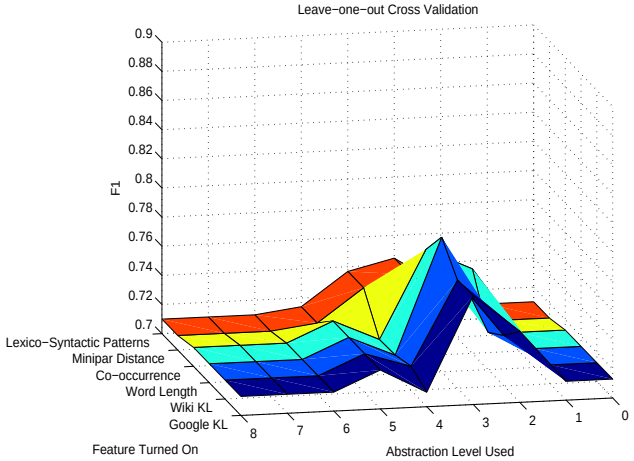

Figure 7: Feature Impact on Different Abstraction Level for ODP

servation that characteristics of abstract levels are general and shared across different ontologies. Very general concepts may even appear in multiple ontologies. On the other hand, concepts at concrete levels are more specific and with few characteristics to share. Moreover, they indicate that the learning of abstract concepts is the most sensitive part in the entire framework - a good modelling on abstract concepts can greatly boost the performance. Given the fact that ontology learning for abstract concepts is harder than that for concrete concepts, our strategy of explicitly modelling the concept abstractness successfully provides the opportunity to improve the weakest link. Further more, for abstract concepts, the contributions from difference features vary, whereas for concrete concepts, the contributions indifferent. This difference confirms the third assumption about ontology space in section 2: different abstraction levels have different characteristics.

Features from six categories (see section 4.2 for more details) are turned on one by one in this experiment. It is interesting to see that simple features, such as term co-occurrence and word length difference, work the best for both datasets. In WordNet datasets, syntactic features, such as lexico-syntactic patterns and minipar distance, works better than contextual features, such as KL-divergence of Google and Wikipedia documents, while in ODP datasets, the conclusion is the opposite. This can be attributed to the fact that WordNet has a richer vocabulary and a more rigid definition of hyponyms and hence more sensitive to linguistic patterns, while ODP contains more noise hence works along better with contextual features generated from the Web.

Lastly, note that experiments shown in Figure 6/7 use the same set of data as in Figure 2/3 when the number of concepts equals 2000. Comparing Figure 6/7 (experiments on individual features) with Figure 2/3 (experiments on all features), we notice a performance gain. It indicates that the combination of heterogenous features gives more rise to the system performance than an individual feature does.

6. CONCLUSIONS

This paper has presented a novel metric-based framework to address three problems in ontology learning. The incremental clustering algorithm avoids and hence solves the *problem of unknown group names* in traditional bottom-up approaches. The framework tackles the *problem of no control over concept abstractness* and explicitly models concept abstractness. The experiments show that concepts at different abstraction levels behave differently and are sensitive to different features. The framework also provides a solution to the *problem of how to incorporate heterogenous semantic features* and the experiments show that it outperforms the use of single feature. The system has been compared to a strong baseline system, and shows an absolute gain of 10% in precision for both WordNet and ODP data. The evaluation demonstrates that the proposed framework is effective for ontology learning.

7. ACKNOWLEDGMENTS

This research was supported by NSF grant IIS-0704210. Any opinions, findings, conclusions, or recommendations expressed in this paper are of the authors, and do not necessarily reflect those of the sponsor.

8. REFERENCES

[1] E. Blomqvist. Fully automatic construction of enterprise ontologies using design patterns: Initial method and first experiences. In *The 5th International Conference on Ontologies, DataBases, and Applications of Semantics*, 2005.

[2] S. Boyd and L. Vandenberghe. Convex optimization. In *Cambridge University Press*, 2004.

[3] S. A. Caraballo. Automatic construction of a hypernym-labeled noun hierarchy from text. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, 1999.

[4] P. Cimiano and J. Vlker. Text2onto: A framework for ontology learning and data-driven change discovery. In *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems*, 2005.

[5] F. Colace, M. D. Santo, and M. Vento. An automatic algorithm for building ontologies from data. In *International Conference on Information and Communication Technologies: From Theory to Applications*, 2004.

[6] D. Crystal. The cambridge encyclopedia of the english language. In *Cambridge: Cambridge University Press*, 1995.

[7] C. Fellbaum. *WordNet:An Electronic Lexical Database*. MIT Press, 1998.

[8] M. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics*, 1992.

[9] L. Karoui, M.-A. Aufaure, and N. Bennacer. Ontology discovery from web pages: Application to tourism. In *In the Workshop of Knowledge Discovery and Ontologies*, 2004.

[10] L. Khan and L. Wang. Automatic ontology derivation using clustering for image classification. In *In Proc. of 8th International Workshop on Multimedia Information Systems*, 2002.

[11] M. Sabou. Extracting ontologies from software documentation. In *Workshop on Ontology Learning and Population, European Conference on Artificial Intelligence*, 2004.

[12] M. Sabou, C. Wroe, C. Goble, and G. Mishne. Learning domain ontologies for web service descriptions: an experiment in bioinformatics. In *Proceedings of the 14th international conference on World Wide Web*, 2005.

[13] M. Sanderson and B. Croft. Deriving concept hierarchies from text. In *Proceedings of the 22nd Annual International ACM SIGIR Conference*, 1999.

[14] R. Snow, D. Jurafsky, and A. Y. Ng. Semantic taxonomy induction from heterogenous evidence. In *COLING/ACL*, 2006.

[15] W. T. Tutte. Graph theory. In *Cambridge University Press*, 2001.

[16] H. Yang and J. Callan. Ontology generation for large email collections. In *Proceedings of the Eighth National Conference on Digital Government Research*, 2008.