PERSONAL ONTOLOGY LEARNING

A THESIS PROPOSAL
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF CARNEGIE MELLON UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Thesis Committee:
Jamie Callan (Carnegie Mellon University, Chair)
Jaime Carbonell (Carnegie Mellon University)
Christos Faloutsos (Carnegie Mellon University)
Eduard Hovy (University of Southern California)

Hui Yang
October 2009

Abstract

This thesis proposal explores Personal Ontology Learning, which is a new task to assist an individual to organize information items, such as documents, web pages, and email messages; to accomplish tasks; and to quickly access information.

This thesis proposal presents two ontology learning frameworks, namely human-guided ontology learning and metric-based ontology learning. Both frameworks combine the strengths of existing pattern-based and clustering-based ontology learning approaches, move beyond the limitation of traditional use of features, and incorporate a wide range of semantic evidence in ontology learning. Each framework provides a unique solution to achieve effective and flexible construction of ontologies.

The two frameworks also differ in various ways with different emphases. The human-guided ontology learning framework addresses personalization and human computer interaction in the process of personal ontology learning. Periodic manual guidance provides training data for learning a distance metric, which is then used during automatic activities to further construct the ontology. A user study demonstrates that human-guided machine learning is able to generate ontologies with manually-built quality and with lower cost. It also shows that periodic manual guidance successfully directs machine learning towards personal preferences. The human-guided ontology learning framework is the first ontology learning framework to construct personalized and task-specific ontologies.

The metric-based ontology learning framework addresses ontology structure optimization and concept abstractness in ontology learning. It is a novel framework which incremental clusters concepts based on minimum evolution of ontology structure and concept abstractness. It is also the first piece of ontology learning work which models abstract concepts and concrete concepts differently in a learning framework to produce more sensible results. An extensive evaluation with WordNet and Open Directory Project data demonstrates that the framework is effective for ontology learning. Moreover, the flexible design of the framework allows a further study of the interaction between features and different types of relations, as well as the interaction between features and concepts with different levels of abstractness.

The research in this thesis proposal is the first step of personal ontology learning, and an important step forward of ontology learning. This work addresses important problems of personal ontology learning, especially considers how to better model these problems with good theoretical foundations, to study these problems via extensive empirical experiments and user studies, and to solve these problems by developing practical applications for the task of personal ontology learning.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Information overload has become a widely recognized problem. Because of the complexity and the pace of today's information-intensive society, there is often too much information to handle and too little time to process and digest the information. Whatever people perceive as information overload affects their decision-making, quality of work, job satisfaction, and leads to frustration, stress, and loss of time.

For example, information overload creates problems in lawsuits. Every year in the state courts, there are millions of filed lawsuits, each of which needs to be properly documented and archived. From piles of transcripts of trial hearings and criminal codes, it is a very challenging task for lawyers to *quickly* access and evaluate *every* detail of a lawsuit process.

Information overload is also a problem in federal rule-making. Due to legal constraints, rule-makers from administrative agencies of the U. S. government are required to seek comments from stakeholders and the public, and respond to substantive issues in the final rule. Each year a few high profile rules attract hundreds of thousands of comments. By law, rule-makers must consider every substantive issue raised during the comment period. When comment volume is high and the time for processing comments is short, evaluating each comment *quickly* and *thoroughly* brings the rule-makers a significant burden.

Besides lawsuits and federal rule-making, information overload also exists in online search activities. Sometimes, the formidable number of search results returned by web search engines do not help Internet users find "needle from the haystack", but exacerbate the problem of information overload. In such cases, Internet users are instantaneously led to an excess amount of information, without knowing the validity of the content and the risk of misinformation.

In all the above situations, knowledge workers, such as lawyers and rule-makers, and average Internet users are overwhelmed by the need to collect every kind of information available and to find technical or daily-life solutions from the largely available information. This need demands a process to archive and organize the related information in order to ease the information access activities and to improve people's capability for inquiry.

This thesis proposal explores a new tool to alleviate the problem of information overload, to gain greater control over the information access experience, and to allow both knowledge workers and Internet users quickly access and evaluate a large amount of information. The new tool that we propose identifies concepts discussed in a set of digitalized documents, and organizes these concepts into an *ontology*.

An *ontology* is an organization of concepts in a domain. *Ontology* is also known as *taxonomy* or *concept hierarchy*. Browsing ontologies, such as the Yahoo! Directory, is a popular method of quickly discovering the "lay of the land" in large text corpora. Moreover, by associating documents with topics and concepts in a hierarchy, ontologies structure the information space and partition the space into smaller spaces; hence ontologies reduce the difficulty in information access. Such a taxonomy of concepts for a text corpus, such as transcripts of trial hearings for a lawsuit, comments on a proposed rule, or search results from a query, will help a knowledge worker or an Internet user more quickly understand the range of the issues raised, and enable "drilling down" into documents that discuss a particular topic.

1.1 Personal Ontology Learning

A common use of ontologies is for knowledge representation. In particular, ontologies are used to explicitly represent, store, and reuse knowledge in a certain domain. By standardizing the terminologies and rules in the domain, ontologies allow knowledge sharing and reuse among multiple users, and may support the making of inferences. Examples of commonly-used ontologies include Library of Congress Subject Headings (LCSH), Medical Subject Headings (MeSH) [NJH01], Gene Ontology [Ash00], WordNet [Fel98], and Cyc [Len95].

Another common use of ontologies is for information organization. This is the focus of this thesis proposal. Since ancient times, hierarchies/ontologies, are commonly used to organize information. In particular, an ontology presents the concepts discussed in a domain, and organizes these concepts based on the relations among them. Concepts are usually topics of interests in a domain. Each concept associates to a set of related documents, which can be accessed quickly because of the direct association between concept and document. Ontologies reveal the relations among concepts and organize the concepts based on these relations; hence ontologies frame the information in the domain in a more organized way. Therefore, ontologies reduce the burden on processing and finding informative materials, and ease the process of information access.

The process of constructing an ontology is called *Ontology Learning*. It usually refers to (semi-) automatic construction and development of ontologies. Although the most reliable ontologies are expected to be manually built by domain experts with an abundance of human expertise, manual construction may be unfeasible in many cases due to time constraints and labor costs.

Ontology learning is concerned with constructing concept hierarchies through accomplishing two subtasks: *concept extraction* and *relation formation*. *Concept extraction* identifies the concepts from a raw document collection. *Relation formation* discovers relations among these concepts and builds an ontology based on these relations.

The goal of ontology learning depends on the use of the ontology. If an ontology is used for knowledge representation, the goal is to explicitly define concepts and relations acquired from text; if an ontology is used for information organization, the goal is to help an individual to quickly understand the information landscape and access specific information. Therefore, for the use of information organization, ontology learning is often required to reflect personal preference in organizing information. The desired organization of information could be based on an individual's own criteria, which can be affected by his personal understanding of the data and the specific information needs or tasks.

Personal ontology learning is the outcome when ontology learning needs to meet personal preferences. It is also the new task proposed and investigated in this thesis proposal. Personal ontology learning assists an individual to organize information items, such as documents, web pages, and email messages; to accomplish tasks; and to fulfill the individual's various requirements. It furnishes an individual with the power of organizing small or massive amount of information to suit his own needs. To aid quick navigation to information and to capitalize on the power of (semi)-automatic organization of the information, we are motivated to explore the task of *personal ontology learning*.

1.2 Challenges

The challenges of *personal ontology learning* rise from both *ontology learning* and *personalization*. A major challenge is to extend existing work on ontology learning as well as to present new solutions to ontology learning. Existing work on ontology learning has been conducted under a variety of names, such as taxonomy induction, semantic class learning, relation acquisition, and relation extraction. The approaches fall into two main categories: *pattern-based* and *clustering-based*. *Pattern-based* approaches define lexical-syntactic patterns for relations, and use these patterns to discover instances of relations. The approaches are known for their high accuracy in discovering relations. However they cannot find relations which do not explicitly appear in text. *Clustering-based* approaches hierarchically cluster terms based on similarities of their meanings usually represented by a vector of features. The approaches complement pattern-based approaches by their ability to discover relations which do not explicitly appear in text. However, they cannot generate relations as accurate as pattern-based approaches can. Ontology learning needs new solutions to extend the existing technologies and combine the strengths of both approaches naturally and flexibly into a unified framework, where the impact of different technologies can be easily investigated and evaluated.

Personalization is another major challenge because it makes ontology learning even more a subjective task. During the process of assisting one to organize information, personal ontology learning must adapt to one's personal understanding of the problem, to one's preference towards a certain aspect of the problem, and to one's purpose of the actual information seeking task. For instance, one may organize "polar bear" and "seal" together since they both are arctic marine mammals; while someone else may organize "polar bear" and "black bear" together since they are both bears. Neither way is wrong; the choice is simply due to different personal criteria. Personal preferences need to be captured during ontology construction and to be reflected in a general learning framework.

A third challenge is handling concept abstractness. Concept abstractness has not been properly addressed in the literature. In an ontology, concrete concepts usually lay at the bottom of the hierarchy while abstract concepts often occupy the intermediate and top levels. Concrete concepts often represent physical entities, such as "basketball" and "mercury pollution". While abstracts concepts, such as "science" and "economy", do not have a physical form thus we must *imagine* their existence. The obvious differences between the two types of concepts suggest that there is a need to treat them differently in ontology learning. Moreover, there are different degrees of abstractness within the abstract concepts, e.g., "science" is more abstract than "computer science". However, most current technologies avoid these issues and simply treat all concepts in the same way with the hope

that the impact of concept abstractness on ontology learning is small, or the different behavior of concrete and abstract concepts can be captured by lexico-syntactic patterns. In this work, we take the challenge and propose to explicitly model concept abstractness in ontology learning.

1.3 Contributions

This thesis proposal presents new techniques for personal ontology learning. In particular, it elaborates two ontology learning frameworks, namely *human-guided* and *metric-based*. Both approaches combine the strengths of existing pattern-based and clustering-based approaches by incorporating lexico-syntactic patterns as one type of feature in a clustering framework. They integrate contextual, co-occurrence, syntactic dependency, lexical-syntactic pattern and other features to learn semantic distances for pairs of concepts, then either *agglomeratively* or *incrementally* cluster concepts based on their distance scores. Both frameworks move beyond the limitation of traditional use of features and incorporate heterogeneous semantic evidence in the learning process. Each framework provides a unique solution to achieve effective and flexible construction of ontologies.

Although both *human-guided* and *metric-based* ontology learning frameworks extend current technologies, incorporate heterogeneous features, and employ machine learning techniques to learn concept distances, the two approaches differ in various ways with different emphasis. The *human-guided* ontology learning framework, presented in Chapter 5, addresses personalization and human computer interaction in the process of personal ontology learning. Periodic manual guidance provides training data for learning a distance metric, which is then used during automatic activities to further construct the ontology. A user study demonstrates that human-guided machine learning is able to generate ontologies with manually-built quality and with lower cost. It also shows that periodic manual guidance successfully directs machine learning towards personal preferences. The *human-guided* ontology learning framework is the first ontology learning framework to construct personalized and task-specific ontologies.

The *metric-based* ontology learning framework, presented in Chapter 6, addresses concept abstractness in ontology learning. It models abstract concepts and concrete concepts differently in the learning framework to produce more sensible results. It also takes an incremental clustering approach to bypass the trouble of labelling non-leaf clusters. The incremental clustering process is transformed into an optimization problem based on two assumptions: *minimum evolution* and *abstractness*. An extensive evaluation with WordNet and Open Directory Project data demonstrates that the framework is effective for ontology learning. Moreover, the flexible design of the framework allows a further study of the interaction between features and different types of relations, as well as the interaction between features and concepts with different levels of abstractness.

1.4 Outline

The following chapters of this proposal are organized as follows. Chapter 2 discusses related work. Chapter 3 describes the datasets used in this thesis proposal. Chapter 4 presents the techniques used for the subtask of concept extraction. Chapter 5 presents the human-guided ontology learning framework. Chapter 6 describes the metric-based ontology learning framework. Chapter 7 presents the future dissertation research, including a

unification of the human-guided and metric-based ontology learning frameworks, a further study on human-task interactions, more extensive user-oriented evaluations, and new methods for automatic cluster labelling. Chapter 8 discusses the expected contributions from the dissertation research. Chapter 9 gives the time estimation for the dissertation. Appendix A shows some example ontologies.

Chapter 2

Related Work

2.1 Ontology Learning

There has been a substantial amount of research on ontology learning. As mentioned earlier, the two main approaches are *pattern-based* and *clustering-based*.

2.1.1 Pattern-based Ontology Learning

Pattern-based approaches are the main trend for ontology learning. The approaches define lexical-syntactic patterns for relations, and use these patterns to discover instances of relations.

For example, one pattern for the *is-a* relation is "*NPx, and other NPy*", where *NPy* indicates hypernym, the parent concept and *NPx* indicates hyponym, the child concept. An exhaustive search of this pattern in a text corpus can discover instances of the *is-a* relation. Each instance contains a pair of noun phrases, filling the positions of *NPx* and *NPy*. If an instance shows that *NPx = 'lawn spray'* and *NPy = 'chemicals'*, it indicates that this pattern finds an instance of the *is-a* relation: *is-a(lawn spray, chemicals)*.

Pattern-based approaches have been applied to extract various types of lexical and semantic relations, including *is-a*, *part-of*, *sibling*, *synonym*, *causal*, and many others.

Pattern-based approaches started from and still pay a great deal of attention to the most common *is-a* relation. Hearst (1992) pioneered using a hand crafted list of hypernym patterns, such as "*NPx, and/or other NPy*" and "*NPy including NPx*", as seeds and employing bootstrapping to discover new instances and patterns for the *is-a* relation [Hea92]. In her work, Hearst manually identified six commonly used lexical-syntactic patterns that indicate the *is-a* relation and used them as the seed patterns. The seed patterns were used to search for instances of the *is-a* relation in text. Each found instance, consisting of a few noun phrases, was used to search environments where the noun phrases occur syntactically near one another. The environments were recorded and the common environments (selected manually) yielded new patterns that indicate the *is-a* relation. The new patterns were then used to discover more instances of the relation. Through this bootstrapping technique, abundant new instances of the *is-a* relation were identified; these instances were successfully used to verify and augment the WordNet noun ontology.

Since Hearst's work, many approaches [Man02, ECD+05, SJN05] have used lexical-syntactic patterns in their work on the *is-a* relation. For instance, Mann (2002) built a fine-grained proper noun ontology from 1GB of AP news wire text to support the task of question answering [Man02]. Mann extracted instances of the *is-a* relation for proper nouns using a single pattern: "*(the)? NPy NPx*", where *NPx* is a proper noun (NNP/NNPS) phrase. This pattern searches for instances of a noun phrase followed by a proper noun phrase. An instance of this pattern is *the automaker Mercedes-Benz*. Mann claimed that this pattern is a more productive pattern than the Hearst patterns for the corpus he used. Similar to the work introduced in [Hea92], Mann also used the acquired instances to augment WordNet and to infer answers for TREC-style factoid questions. He further extended the pattern's coverage by inferring new instances through a simple rule and existing instances. The rule is "*is-a(NPa, NPy) and is-a(NPb, NPy) and is-a(NPb, NPz)* $\Longrightarrow$ *is-a(NPa, NPz)*". For example, if we know instances *is-a(Mercedes-Benz, automaker) and is-a(Opel, automaker) and is-a(Opel, car company)*, we can infer a new instance, *is-a(Mercedes-Benz, car company)*. Such inference rule is able to extend the coverage of lexical-syntactic patterns, however, as pointed out by Mann himself, care must be taken to ensure the inferences are proper. However, Mann did not discuss how to ensure proper inferencing in this paper.

Pantel, Ravichandran, and Hovy (2004) also worked on mining instances of the *is-a* relation [PRH04]. Unlike Hearst's work, which identified new patterns by hand, Pantel, Ravichandran, and Hovy (2004) automatically identified new *is-a* patterns by a minimal edit distance algorithm, which selected the optimal common patterns based on the overall costs associated with each string insertion and deletion among similar instances. For example, for sentence "*Platinum is a precious metal*", POS-tagged as "*NNP VBZ DT JJ NN*", and sentence "*Molybdenum is a metal*", POS-tagged as "*NNP VBZ DT NN*", the optimal common pattern is "*NNP is a (*s*) metal*", where *(*s*)* is a skip wildcard operator. The authors also applied the patterns to a 15GB corpus, where they first tested lexical-syntactic patterns for the *is-a* relation acquisition at the terascale.

Another common relation is *sibling*, which describes the relation of sharing similar meanings and being members of the same class. Words in *sibling* relations are also known as *class members* or *similar words*.

Inspired by the observation that words are often surrounded by sibling words in text, for example in conjunctions (*lions, tigers, and bears*), lists (*lions, tigers, bears ...*), appositives (*the stallion, a white Arabian*), and nominal compounds (*Arabian stallion; tuna fish*), Riloff and Shepherd (1997) [RS97] used co-occurrence statistics in local context to discover instances of the *sibling* relation for nouns. Their work used a small set of seed words for a class, and identified sibling words that also belong to the class. In particular, a narrow context window consisting of only the left first noun and the right first noun to each seed word was collected. The authors reported that such a narrow context window performed better than a larger context window for this task. The top five nouns, which were not already seed words and showed high conditional probability within the context windows of a certain class over the entire corpus, were selected as new seed words, i.e., new instances of siblings. The authors reported that an eight-iteration bootstrapping worked well. Although this work did not directly use lexical-syntactic patterns to mine instances, it used narrow context windows to mimic patterns representing conjunctions, lists, appositives, and nominal compounds. Therefore, this work is still generally considered as a pattern-based approach. A limitation of this work is that it only handled nouns, not noun phrases.

Roark and Charniak (1998) [RC98] also used co-occurrence statistics in local context to discover instances of the *sibling* relation. Their work extended Riloff and Shepherd (1997)'s work to noun phrases by only counting

word co-occurrence within a noun phrase, between head nouns that are separated by a comma or conjunction. For example, for sentence "*A cargo aircraft, fighter plane, or combat helicopter ...*", only *aircraft*, *helicopter*, and *plane* would be counted as co-occurring with each other in this work. In contrast, in Riloff and Shepherd (1997)'s work, *cargo*, *aircraft*, and *fighter* would be counted as co-occurring with each other; the calculation was incorrect in this case. Moreover, Roark and Charniak (1998) tried to distinguish the nominal compounds whose non-head noun is a legitimate member of a class versus the nominal compounds whose non-head noun is not a legitimate member of a class. Only the former kind of nominal compounds were selected as candidates for new seeds. The authors made this distinction by setting a cutoff threshold on the ratio of the number of times a noun occurs as a non-head noun in any nominal compound and the number of times this noun occurs in any nominal compound. If the ratio for a non-head noun in a nominal compound was above the threshold, this non-head noun was kept as a candidate for new seeds for the class indicated by the head noun. For instance, in *fighter plane* the non-head noun *fighter* is a hyponym of *plane*; whereas in *government plane* the non-head noun *government* is not a hyponym of *plane*.

The KnowItAll system [ECD$^+$05] extended the work in [Hea92] and bootstrapped lexical-syntactic patterns on the Web to discover siblings; the system also ranked and selected the patterns by statistical measures.

Widdows and Dorow (2002) combined symmetric patterns, such as "*NPx and/or NPy*", and graph link analysis to discover sibling relations [WD02]. Davidov and Rappoport (2006) also used symmetric patterns for identifying sibling relations [DR06].

Recently, Kozareva, Riloff and Hovy (2008) combined a single double-anchored hyponym pattern, "*NPy such as NPx and **", with graph structure to extract siblings [KRH08].

The third common relation is *part-of*. Berland and Charniak (1999) used two meronym patterns, "*NPy's NPx*" and "*NPx of a/an NPy*" to discover *part-of* relations, and also used statistical measures to rank and select the matching instances [BC99]. Girju et al. (2003) [GBM03] took a bootstrapping approach similar to Hearst (1992) for *part-of* relations.

Other types of relations that have been studied by pattern-based approaches include question-answer relations (such as *birthdates*, *why-famous*, and *inventor*) by Ravichandran and Hovy (2004) [RH02], synonyms and antonyms by Lin et al. (2003) [LZQZ03], general purpose analogy by Turney et al. (2003) [TLBS03], verb relations (including *similarity*, *strength*, *antonym*, *enablement* and *temporal*) by Chklovski and Pantel (2004) [CP04], entailment by Szpektor et al. (2004) [SHTC04], and more specific relations, such as *consist-of*, *purpose*, *creation*, and *completion* by Cimiano and Wenderoth (2007) [CW07], and *LivesIn(Person, Location)*, *EmployedBy(Person, Organization)*, and *CorpAcquired(Organization, Organization)* by Bunescu and Mooney [BM07].

Pattern-based approaches are known for their high accuracy in recognizing instances of relations if the patterns are carefully chosen, either manually [BC99, KRH08] or via automatic bootstrapping [Hea92, ECD$^+$05, GBM03, RH02, PP06].

The most commonly used technique in pattern-based approaches is *bootstrapping* [Hea92, ECD$^+$05, GBM03, RH02, PP06]. It utilizes a few man-crafted seed patterns to extract matching instances from corpora, then extracts new patterns using these instances, and continues the cycle to find new instances and new patterns. Bootstrapping is effective and scalable to large datasets; however, uncontrolled bootstrapping soon generates undesired instances once a noisy pattern is brought into the cycle.

To aid bootstrapping, methods of pattern quality control are widely applied. Statistical measures, such as point-wise mutual information [ECD+05, PP06] and conditional probability [CW07], have been shown to be effective to rank and select patterns and instances. Pattern quality control is also investigated by using WordNet [GBM06], graph structures built among terms [WD02, KRH08], clusters of similar patterns [DR06], and topic modelling for multiple membership [HZ09].

Although pattern-based approaches are able to produce highly accurate instances of relations, the approaches suffer from sparse coverage of patterns in a given corpus. Recent studies [ECD+05, KRH08] show that if the size of a corpus, such as the Web, is nearly unlimited, a pattern has a higher chance to explicitly appear in the corpus. However, corpus size is often not that large; hence the problem still exists. Moreover, since patterns usually extract instances in pairs, the approaches suffer from the problem of inconsistent concept chains after connecting pairs of instances to form taxonomy hierarchies. For instance, *car* has hyponyms of *coupe*, *sedan* and *sport*, *sport* has *basketball*. It is clear that *car* should not be ascendent of *basketball*.

2.1.2 Clustering-based Ontology Learning

Clustering-based approaches usually represent word contexts as vectors and hierarchically cluster words based on similarities of the vectors [BPd+92, Lin98]. Clustering-based approaches have only been applied to solve *is-a* and *sibling* relations.

Based on the clustering algorithms used, clustering-based approaches can be divided into two groups: agglomerative and divisive. Agglomerative clustering [BPd+92, Car99, DH02, KW02, RF07, YC08a] iteratively merges the most similar clusters into bigger clusters. Lin (1998) defined a similarity measure between two words based on each word's number of occurrences in syntactic dependency triples [Lin98]. A syntactic dependency triple consists of two words and their grammatical relationship. For example, a triple "brown adj-mod-of dog" can be extracted from the sentence "I have a brown dog". Lin (1998) then used agglomerative clustering to create thesauri. Caraballo (1999) represented a word's context as a vector containing counts of each other word appearing in conjunction or appositive with this word [Car99]; Caraballo (1999) then agglomeratively clustered words based on cosine similarity between the vectors. Recently, Rosenfeld and Feldman (2007) used surface patterns as features, selected the features by statistical measures, and then used agglomerative clustering based on cosine similarity between the feature vectors.

Another types of clustering algorithm is divisive clustering. Divisive clustering iteratively splits clusters into smaller clusters using a partition clustering algorithm, such as K-means [Mac67]. This procedure is applied recursively until each word is in its own singleton cluster. Cimiano et al. [CHS04] used bi-section k-means divisive clustering. CBC (Clustering By Committee) is an example of such a divisive clustering algorithm [PL02, PR04], which calculates cluster centroids by averaging the feature vectors of a subset of carefully chosen cluster members, and then iteratively splits clusters.

More work on clustering-based ontology learning is described in [BCM05, CV05a].

A main advantage of clustering-based approaches is that they are able to discover relations which do not explicitly appear in text. This is because they do not require a concept explicitly appear in text to match with a pattern; as long as the context of the concept is similar to the context of other concepts, a relation, such as sibling, can be identified. Clustering-based approaches also avoid the problem of inconsistent chains by addressing the

structure of a taxonomy globally from the outset.

Nevertheless, it was pointed out by researchers that clustering-based approaches cannot generate relations as accurately as pattern-based approaches. Pantel and Pennacchiotti (2006) revealed that clustering-based approaches generally fail to produce coherent cluster for small corpora [PP06]. Moreover, the performance of clustering-based approaches is largely influenced by the types of features used. The common types of features include contextual features [Lin98], verb-noun relations [FPL93], syntactic dependency [PR04, SJN05], surface patterns [RF07], co-occurrence [YC08a, YC09b, YC09a], conjunction and appositive features [Car99]. So far, there is no systematic study of which features are the best for automatic ontology learning under various conditions.

Many clustering-based approaches also face the challenge of appropriately labeling non-leaf clusters. The labeling amplifies the difficulty in the creation and the evaluation of ontologies. In agglomerative clustering, clusters are merged into bigger clusters, which need to be labeled; in divisive clustering, parents of split clusters also need to be labeled.

An incremental clustering approach is promising for avoiding labeling clusters. In Chapter 6, we present such an approach, in which terms and relations are added into an ontology one at a time, and their parents come from the existing ontology. The advantage of the incremental approach is that it eliminates the trouble of inventing cluster labels and concentrates on placing terms in the correct positions in a concept hierarchy.

The work by Snow et al. (2006) [SJN06] is the most similar to our work in Chapter 6 because it also took an incremental approach to construct ontologies. In their work, an ontology grows based on maximization of conditional probability of relations given evidence; while in our work it grows based on optimization of ontology structures and modeling of term abstractness. Moreover, our approach employs heterogeneous features from a wide range while the prior work used only syntactic dependency. We compare system performance between [SJN06] and our metric-based ontology learning framework in Chapter 6.

Sanderson and Croft [SC99] identifies a term's hyponyms and hypernyms by studying the ratio of document frequencies of two terms. In their approach, a term x is term y's parent in the ontology hierarchy (i.e., hypernym)if the conditional probability of y given x $p(x|y) \geq 0.8$ and conditional probability of x given y $p(y|x) < 1$. Lawrie and Croft [LC03] continued their work by employing language models to develop ontologies as document summaries.

Recently, Cimiano et al. applied Formal Concept Analysis [CHS04] to construct concept object and feature lattice, and then look for common features shared by different objects as the super-concepts for them.

2.2 Human-Guided Machine Learning

Human-guided machine learning is an emerging topic which unites human computer interaction (HCI) and machine learning (ML). It is the study of designing and developing machine learning algorithms which periodically receive manual guidance in a complete human-teaching-machine-learning loop. Inspired by the Situated Learning Theory in cognitive science [Gre84], human-guided machine learning provides a framework for real-time learning from interactions with humans, and studies the effects of periodic manual guidance on the learning process.

Human-guided machine learning is related to Active Learning [CGJ96, SC00]. In active learning, a machine learning algorithm *actively* queries human users for data labels in which the algorithm has low confidence. Human-guided machine learning and active learning are similar in terms of the human involvement in machine learning,

and the loop of human computer interaction. However, they differ in whether the human or the machine leads the interaction. In active learning, a machine learning algorithm is in control of the interaction and actively requests data labels from human users, ignoring what they want to provide. In contrast, in human-guided machine learning, the human leads the interaction and actively provides guidance to the machine learning algorithm via data labeling or other methods; the machine learning algorithm does not actively choose data samples for the human to label. Moreover, human-guided machine learning and active learning are different in their goals. Active learning aims to save human labelling effort since the learning algorithm is able to use abundant unlabeled data and only requests manual labels for data with low confidence. In contrast, human-guided machine learning aims to design learning algorithms which follow human directions in real-time and produce results reaching human satisfaction.

While significant attention has been paid to developing machine learning algorithms trained by data provided all at once and which perform well on their own, much less attention has been paid to developing human-guided machine learning algorithms. To date, human-guided machine learning has only been studied in the subfields of machine learning, including robot task learning [NM03, CV08], reinforcement learning [MST$^+$05, TB06], classification [SK01, TC09], and clustering [KKSM05, HM07].

Among these subfields, human-guided clustering is the subfield which is most related to personal ontology learning. Kerne et al. (2005) organized information elements in a spatial hypertext space which a user could adjust by choosing cluster anchors and setting relative importance of features through a pie chart [KKSM05]. Huang and Mitchell (2007) defined the two-way-communication between the human and the machine in mixed-initiative clustering [HM07, HM08]. During the communication from the machine to the human, the SpeClustering algorithm [HM06] presented a list of properties, including labels of instances, must-links among instances, cluster existence, and key features, to the human. During the communication from the human to the machine, the human then provided feedback, such as approval, disapproval, modification, and introduction of new properties, to the machine. The machine took the human feedback as true values or constraints to these properties, and re-trained the clustering algorithm by adjusting probabilities according to the feedback.

Personal ontology learning requires personalization during ontology construction. The personal preferences are usually collected via human computer interaction. In Chapter 5, we develop a human-guided ontology learning framework, which incorporates manual guidance during the ontology construction process. The manual guidance is implicitly presented by a partial human-constructed ontology. The learning algorithm takes the partial human-constructed ontology as training data and learns a distance function between concepts; the algorithm then applies the newly-learned distance function to unclustered concepts. The loop continues until human satisfaction.

Chapter 3

Datasets

This chapter describes the set of text collections used to perform the experiments in this thesis proposal. The datasets include public comments datasets, WordNet ontologies, ODP ontologies, and auxiliary datasets.

3.1 Public Comments

U.S. law defines a process known as *Notice and Comment Rulemaking* that requires regulatory agencies to seek comment from the public before establishing new regulations. Most proposed regulations attract few comments from the public, but each year a few regulations attract tens of thousands or hundreds of thousands of comments. The comments can be in various formats, including emails, scanned images of mailed letters, faxed documents, etc. Since there is unavoidable recognition lost in converting scanned image to text, only text emails are used in the experiments reported here.

Public comments usually include many comments that are created based on form letters written by special interest groups, such as *BushGreenWatch*, which claims to "track the Bush administrations environmental misdeeds". Modifying an electronic form letter is very easy, and hence many people have begun doing so to better represent their opinions. As a result, public comment datasets often contain duplicated or near-duplicated comments. In this work, we use near-duplicate detection techniques [YC06] to remove the near-duplicates. After duplicate detection, there are still tens of thousands of unique comments.

Table 3.1: Statistics of Public Comment Datasets.

Statistic	**TRI**	**Wolf**	**Polar Bear**	**Mercury**
#comments	86,763	282,992	624,947	536,975
#unique comments	16,367	59,109	73,989	104,146
#words	18,621,800	64,048,109	131,103,438	104,564,253
#words after duplicate removal	141,063	707,515	472,366	6,327,563
#vocabulary	6,582	27,742	16,346	31,975
#concepts	248	795	351	1,084

Four public comment data sets are used for the experiments in this thesis proposal, namely "Toxic Release Inventory (TRI)" (Docket id: USEPA-TRI-2005-0073),"Wolf" (USFWS-R6-ES-2008-0008), "Polar Bear" (USDOI-FWS-2007-0008) and "Mercury" (USEPA-OAR-2002-0056).

The TRI dataset was provided by the U.S. Environmental Protection Agency (EPA) in 2006. These public comments are about a rule that EPA proposed to revise certain requirements for the Toxics Release Inventory to reduce reporting burden while continuing to provide valuable information to the public.

The Wolf dataset was provided by the U.S. Department of Interior, Fish and Wildlife Service (FWS) in 2008. These public comments are about a rule that the Fish and Wildlife Service proposed to designate the northern Rocky Mountain population of gray wolf as a distinct population segment, and to remove gray wolf from the federal list of endangered and threatened wildlife.

The Polar Bear dateset was provided by U.S. Department of Interior, Fish and Wildlife Service in 2007. These public comments are about a rule proposed by the Fish and Wildlife Service to list the polar bears as a threatened species under the Endangered Species Act and to initiate scientific review to study the current situation and future of polar bears.

The Mercury dataset was provided by EPA in 2004. These public comments are about the proposed national emission standards for hazardous air pollutants and about, in the alternative, the proposed standards of performance for new and existing stationary sources: electric utility steam generating units.

Table 3.1 shows the number of comments, the number of comments after duplicate detection (unique comments), the total number of words, the total number of words after duplicate detection, the size of the vocabulary, and the number of concept candidates in these datasets. Concept candidates are obtained by the techniques presented in Chapter 4.

3.2 WordNet and ODP Ontologies

WordNet and Open Directory Project (ODP) are two available human-built ontologies. Due to computational limitation, we do not attempt to reconstruct the entire WordNet or ODP. Instead, we only extract some branches/subdirectories from WordNet and ODP; each of these branches/subdirectories is used as an ontology. Each ontology becomes one dataset in our experiments. These datasets are used to investigate our algorithms' ability to reconstruct concept hierarchies.

We extract hypernym ontologies from both WordNet and ODP, and meronym ontologies from WordNet only (There is no meronym ontology available in ODP). The hypernym ontologies indicate *is-a* relation among concepts; the meronym ontologies indicate *part-of* relation among concepts.

In WordNet ontology extraction, we only use the word senses within a particular ontology to ensure no ambiguity. Therefore there is no multiple senses for a word in one ontology.

In ODP ontology extraction, we parse the topic lines, such as "Topic r:id='Top/Arts/Movies'", in the XML databases to obtain relations, such as *is-a(movies, arts)*.

In total, there are 100 hypernym ontologies, 50 each extracted from WordNet and ODP; and 50 meronym ontologies from WordNet. Table 3.2 shows the ontology statistics, including the number of datasets for each type of ontology, the total, average and maximum number of concepts for each type of ontology, the average and

Table 3.2: Statistics of WordNet and ODP Datasets.

Statistic	WordNet/*is-a*	ODP/*is-a*	WordNet/*part-of*
total #datasets	50	50	50
total #concepts	1,964	2,210	1,812
average #concepts	39	44	37
max #concepts	86	104	79
average depth	5.5	5.9	4.9
max depth	9	8	7
the level w/ most concepts	4	4	3

maximum number of depth of the ontologies, and the id of the level that containing that most number of concepts for each type of ontology.

The 50 WordNet hypernym ontologies are from 12 topics: *gathering, professional, people, building, place, milk, meal, water, beverage, alcohol, dish,* and *herb.*

The 50 ODP hypernym ontologies 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.*

The 50 WordNet meronym ontologies are from 15 topics: *bed, car, building, lamp, earth, television, body, drama, theatre, water, airplane, piano, book, computer,* and *watch.*

3.3 Auxiliary Datasets

We also use two Web-based auxiliary datasets to generate semantic features mentioned in Section 5.5. When the corpus of interest is given for ontology learning, the auxiliary datasets provide additional context for the concept that we would like to study; when only concepts are given and no corpus is given for ontology learning, the auxiliary datasets are the only source to obtain contextual information for concepts. The two auxiliary datasets are:

Wikipedia auxiliary dataset: The entire Wikipedia corpus was downloaded and indexed by the Indri search engine [SMTC05]. This auxiliary dataset is a collection of the top 100 documents returned by Indri when querying with each concept or each concept pair.

Google auxiliary dataset: This auxiliary dataset is a collection of the top 1,000 documents returned by the Google web search engine when querying Google with each concept or each concept pair.

Both auxiliary datasets are split into sentences, and parsed by a POS tagger[1], a semantic role tagger (Assert[2]), and a syntactic parser (Minipar[3]). These datasets are used to generate contextual, co-occurrence, syntactic dependency and lexico-syntactic pattern features described in Section 5.5. In particular, both datasets can be used to generate *global context KL-divergence, local context KL-divergence, document PMI, sentence PMI, Minipar syntactic distance, modifier/object/subject/verb overlap,* and *hypernym/sibling/part-of pattern-based features.* Only Google auxiliary dataset is used to generate *Google PMI.*

[1] http://nlp.stanford.edu/software/tagger.shtml.
[2] http://cemantix.org/assert.
[3] http://www.cs.ualberta.ca/lindek/minipar.htm.

Chapter 4

Concept Extraction

There are two main subtasks in personal ontology learning, namely *concept extraction* and *relation formation*. This chapter details our techniques for *concept extraction*.

Concept extraction is the process of identifying concepts from a raw document collection. The raw document collection can be web pages retrieved by search engines, documents collected during legal discovery, or public comments. Collection sizes can range from a few hundred documents to hundreds of millions of documents. Concepts usually are nouns or noun phrases, which are topics of interest in a given corpus. Sometimes, concepts with the same meaning, i.e. synonyms, are identified and consolidated into a single entry. However, in this work, we assume uniqueness of each concept and do not unify the synonyms. *Concept exaction* is a relatively easy task. Part-of-speech (POS) tagging, verb predicate extraction [Sab04, WVH06], and topic centroid extraction [FMG05] are common techniques to extract concepts from text.

In our work, there are two main steps in the subtask of *concept extraction*, namely *concept mining* and *concept filtering*.

4.1 Concept Mining

Concept mining aims to extract all possible concept candidates from a corpus. It emphasizes concept recall. Our strategy is to extract nominal unigrams, bigrams, trigrams and named entities (NE) as concept candidates at this stage.

To obtain nominal unigrams, bigrams and trigrams, each document in a corpus is split into sentences; and each sentence is parsed by a part-of-speech (POS) tagger [1]. The POS tagger annotates each word with its linguistic part-of-speech. For instance, the sentence "*I strongly urge you to cut mercury emissions from power plants by 90 percent by 2008.*" is tagged as shown below:

I/PRP strongly/RB urge/VBP you/PRP to/TO cut/VB mercury/NN emissions/NNS from/IN power/NN plants/NNS by/IN 90/CD percent/NN by/IN 2008/CD ./.

A noun sequence generator then scans through the parsed sentence to identify noun sequences, i.e., sequences

[1] http://nlp.stanford.edu/software/tagger.shtml

Mercury Dataset	Polar Bear Dataset
#unique bigrams: 462	#unique bigrams: 1,389
total # of bigrams: 4,510	total # of bigrams: 3,397
power plants 370	greenhouse gas 248
mercury pollution 250	gas pollution 227
mercury emissions 175	sea ice 143
mercury levels 110	ice habitat 117
clear air 70	endangered species 115
#unique trigrams: 129	#unique trigrams: 361
total # of trigrams: 900	total # of trigrams: 731
clean air act 65	greenhouse gas pollution 227
environmental protection agency 35	sea ice habitat 115
air quality rule 30	endangered species act 104
pollution control technology 25	arctic sea ice 62
interstate air quality 25	greenhouse gas emissions 19

Table 4.1: Top Bigrams and Trigrams.

of words tagged only with NN (singular noun), NNP (proper noun), NNS (plural nouns), or NNPS (plural proper nouns). In this example, *mercury/NN emissions/NNS* and *power/NN plants/NNS* are two such noun sequences. As a result, the noun sequence generator produces a collection of noun sequences with various lengths.

An n-gram generator then scans through this collection of noun sequences and produces 4 lists of n-grams, namely unigrams, bigrams, trigrams, and longer noun sequences (length>3); the n-gram generator also ranks each list of n-grams by frequency of occurrence in the descending order. Table 4.1 shows the top bigrams and trigrams generated from the Mercury and the Polar Bear public comment datasets (Chapter 3).

Besides nominal unigrams, nominal bigrams, and nominal trigrams, named entities are also good concept candidates. Though we can easily use commercial NE taggers like BBN's Identifier to find named entities, in this work we develop a simple NE tagger for its simplicity, speed, and low cost. Our simple NE tagger only uses the longer noun sequences as input since shorter noun sequences, unigrams, bigrams and trigrams, are already included as concept candidates. The simple NE tagger tags a longer noun sequence as a named entity if the sequence capitalizes the first letters in every word. For example, the noun sequence "*Toxics/NNP Release/NNP Inventory/NNP Burden/NNP Reduction/NNP Proposed/NNP Rule/NNP*" is tagged as a named entity.

These nominal unigrams, nominal bigrams, nominal trigrams, and simple named entities are the initial concept candidates identified by *concept mining*.

Table 4.2 shows the performance of concept mining for four public comment datasets. (Details of the datasets are presented in Chapter 3.) The ground truth of concept extraction is generated by extracting concepts from ontologies constructed by human annotators; the manual ontology construction process is shown in Section 5.7. The initial concept candidates are compared with the ground truth in terms of Precision, Recall and F1-measures. Precision (P) is calculated as the number of correctly returned concept candidates divided by the number of returned concept candidates. Recall (R) is calculated as the number of correctly returned concept candidates divided by the total number of correct concept candidates in the ground truth. F1-measure is calculated as 2*P*R/(P+R). Bold font indicates the best performance in a column.

Table 4.2 also shows the performance of concept mining at different cut-off thresholds of frequency. As the

Table 4.2: Performance of Concept Mining.

Dataset	P(Freq.≥1)	R(Freq.≥1)	F1(Freq.≥1)
Tri	0.03	**0.55**	0.05
Wolf	0.03	**0.75**	0.06
Polar Bear	0.02	**0.70**	0.04
Mercury	0.02	**0.59**	0.03
	P(Freq.≥2)	**R(Freq.≥2)**	**F1(Freq.≥2)**
Tri	0.33	0.54	0.41
Wolf	0.43	0.71	0.53
Polar Bear	0.42	0.68	0.52
Mercury	0.38	0.57	0.47
	P(Freq.≥5)	**R(Freq.≥5)**	**F1(Freq.≥5)**
Tri	**0.53**	0.52	**0.53**
Wolf	**0.63**	0.71	**0.63**
Polar Bear	**0.62**	0.67	**0.62**
Mercury	**0.68**	0.56	**0.68**

Table 4.3: Performance After Concept Filtering.

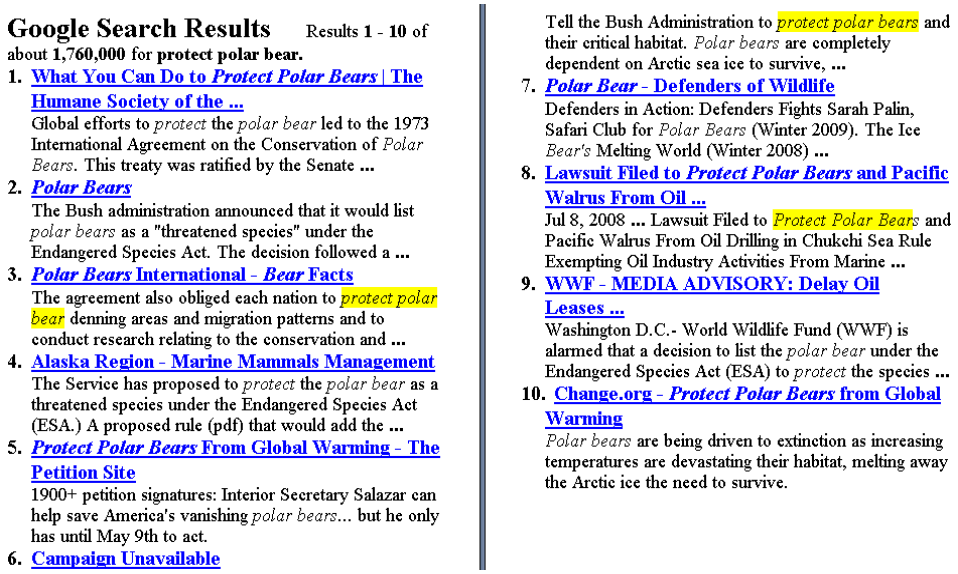
Dataset	P	R	F1	F1 Improvement
Tri	0.76	0.52	0.65	21%
Wolf	0.86	0.71	0.73	15%
Polar Bear	0.80	0.67	0.68	10%
Mercury	0.93	0.56	0.79	16%

cut-off threshold increases to a higher frequency, fewer concepts are kept, Precision increases, Recall slightly drops, and F1 increases. Note that when the cut-off threshold changes from 1 to 2, Precision and F1 increase significantly. This result suggests that concept candidates appearing only once are basically noise. When the cut-off threshold is set to frequency>=5, Precision of concept mining ranges from 0.53 to 0.68, Recall ranges from 0.52 to 0.56, and F1 ranges from 0.53 to 0.58. These results are moderately good, but not good enough for constructing the ontology.

4.2 Concept Filtering

Concept filtering removes errors generated during concept mining. In concept mining, a part-of-speech tagger is employed to recognize nouns. However, POS taggers make mistakes. There are even more tagging errors when documents have spelling, capitalization, punctuation and grammar errors, which are common in public comments and emails. POS tagging error, especially the false positive error, is one of the major error sources in *concept mining*. For example, *protect polar bear* is incorrectly tagged by the POS tagger as three nouns, and hence the n-gram generator incorrectly outputs it as a nominal trigram.

We observe that many false positive noun sequences consist of a non-noun, either a verb or an adjective, attached before or after a noun phrase. For example, *protect polar bear* consists of a verb before a noun phrase; *mercury pollution kills* consists of a verb after a noun phrase. The noun phrase is often the desired concept, which



Google Search Results Results **1** - **10** of about **1,760,000** for **protect polar bear.**

1. **What You Can Do to *Protect Polar Bears* | The Humane Society of the ...**
   Global efforts to *protect* the *polar bear* led to the 1973 International Agreement on the Conservation of *Polar Bears*. This treaty was ratified by the Senate ...
2. ***Polar Bears***
   The Bush administration announced that it would list *polar bears* as a "threatened species" under the Endangered Species Act. The decision followed a ...
3. ***Polar Bears* International - *Bear* Facts**
   The agreement also obliged each nation to *protect polar bear* denning areas and migration patterns and to conduct research relating to the conservation and ...
4. **Alaska Region - Marine Mammals Management**
   The Service has proposed to *protect* the *polar bear* as a threatened species under the Endangered Species Act (ESA.) A proposed rule (pdf) that would add the ...
5. ***Protect Polar Bears* From Global Warming - The Petition Site**
   1900+ petition signatures: Interior Secretary Salazar can help save America's vanishing *polar bears*... but he only has until May 9th to act.
6. **Campaign Unavailable**
   Tell the Bush Administration to *protect polar bears* and their critical habitat. *Polar bears* are completely dependent on Arctic sea ice to survive, ...
7. ***Polar Bear* - Defenders of Wildlife**
   Defenders in Action: Defenders Fights Sarah Palin, Safari Club for *Polar Bears* (Winter 2009). The Ice *Bear's* Melting World (Winter 2008) ...
8. **Lawsuit Filed to *Protect Polar Bears* and Pacific Walrus From Oil ...**
   Jul 8, 2008 ... Lawsuit Filed to *Protect Polar Bears* and Pacific Walrus From Oil Drilling in Chukchi Sea Rule Exempting Oil Industry Activities From Marine ...
9. **WWF - MEDIA ADVISORY: Delay Oil Leases ...**
   Washington D.C.- World Wildlife Fund (WWF) is alarmed that a decision to list the *polar bear* under the Endangered Species Act (ESA) to *protect* the species ...
10. **Change.org - *Protect Polar Bears* from Global Warming**
    Polar bears are being driven to extinction as increasing temperatures are devastating their habitat, melting away the Arctic ice the need to survive.

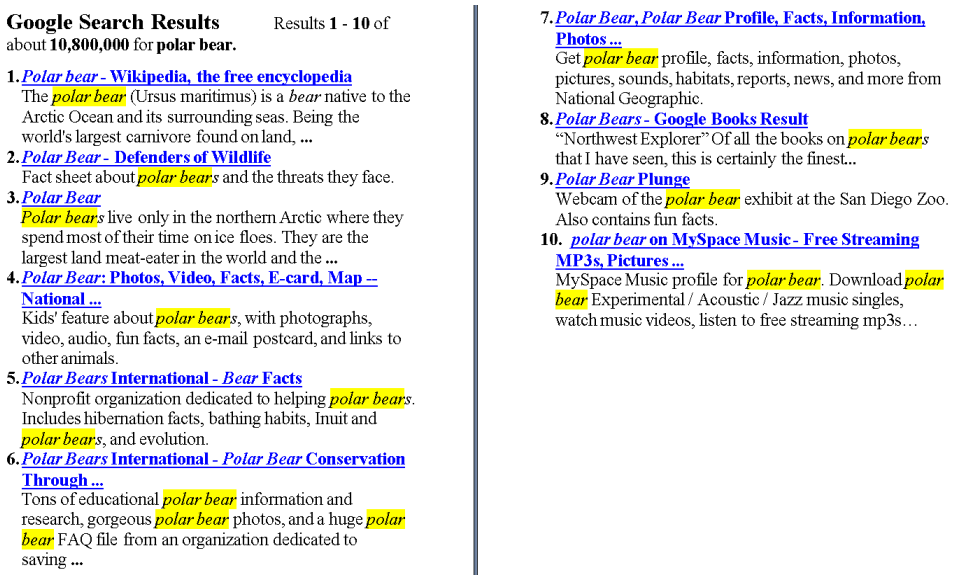
Figure 4.1: Google snippets for *protect polar bear*: 3 occurrences (an error).

usually occurs frequently in daily language. Conversely, the joint occurrence of the verb/adjective and the noun phrase is usually much less frequent.

Based on the above observations, we designed a web-based frequency test to filter out false positive concept candidates. We used the Google search results to evaluate concept candidates. Queries formed by each concept candidate are sent to Google search engine. For example, a query "polar bear" was formed for the concept candidate "polar bear". Among the texts of the first 10 returned Google snippets (not including the titles nor the webpage addresses), if a concept candidate appears more than a threshold number of times (set to 4 in this work), the candidate is considered as a commonly-used phrase, and hence a good concept candidate; otherwise, an error.

Figure 4.1 shows the snapshot of the first 10 Google search results for *protect polar bear*. Since the occurrences of this phrase is only 3 (< 4), our web-based frequency test judges that this phrase is not a commonly-used noun phrase, hence it is not a valid concept candidate and is filtered out. Figure 4.2 shows the snapshot of the first 10 Google search results for *polar bear*. Since the occurrences of this phrase is 14 (> 4), our web-based frequency test judges that this phrase is a commonly-used noun phrase, hence it is a valid concept candidate and is kept.

In this way, false positive concept candidates are effectively identified and removed. Moreover, spelling errors, such as *polor bear*, *pulution*, are also removed. Table 4.3 shows Precision, Recall and F1 after concept filtering. The Precision values range from 0.76 to 0.93, Recall values range from 0.52 to 0.67, F1-measures range from 0.65 to 0.79, with a 10%-21% improvement over concept mining without filtering. These results demonstrate that concept filtering is effective to remove tagging errors and to improve the overall concept extraction performance.



Google Search Results Results **1** - **10** of about **10,800,000** for **polar bear.**

1. *Polar bear* - **Wikipedia, the free encyclopedia**
   The *polar bear* (Ursus maritimus) is a *bear* native to the Arctic Ocean and its surrounding seas. Being the world's largest carnivore found on land, **...**
2. *Polar Bear* - **Defenders of Wildlife**
   Fact sheet about *polar bear*s and the threats they face.
3. *Polar Bear*
   *Polar bear*s live only in the northern Arctic where they spend most of their time on ice floes. They are the largest land meat-eater in the world and the **...**
4. *Polar Bear*: **Photos, Video, Facts, E-card, Map -- National ...**
   Kids' feature about *polar bear*s, with photographs, video, audio, fun facts, an e-mail postcard, and links to other animals.
5. *Polar Bears* **International** - *Bear* **Facts**
   Nonprofit organization dedicated to helping *polar bear*s. Includes hibernation facts, bathing habits, Inuit and *polar bear*s, and evolution.
6. *Polar Bears* **International** - *Polar Bear* **Conservation Through ...**
   Tons of educational *polar bear* information and research, gorgeous *polar bear* photos, and a huge *polar bear* FAQ file from an organization dedicated to saving **...**
7. *Polar Bear*, *Polar Bear* **Profile, Facts, Information, Photos ...**
   Get *polar bear* profile, facts, information, photos, pictures, sounds, habitats, reports, news, and more from National Geographic.
8. *Polar Bears* - **Google Books Result**
   "Northwest Explorer" Of all the books on *polar bear*s that I have seen, this is certainly the finest**...**
9. *Polar Bear* **Plunge**
   Webcam of the *polar bear* exhibit at the San Diego Zoo. Also contains fun facts.
10. *polar bear* **on MySpace Music - Free Streaming MP3s, Pictures ...**
    MySpace Music profile for *polar bear*. Download *polar bear* Experimental / Acoustic / Jazz music singles, watch music videos, listen to free streaming mp3s…

Figure 4.2: Google snippets for *polar bear*: 14 occurrences (a noun phrase).

4.3 Summary

This chapter details our techniques in concept extraction. Concept extraction is performed in two steps: concept mining and concept filtering. Concept mining examines a given corpus and outputs all possible concept candidates within the corpus. The possible concept candidates are nominal unigrams, bigrams, and trigrams, and named entities. Concept filtering further examines the candidates' validity by conducting a web-based frequency test. Concept filtering effectively removes tagging errors produced in concept mining. The concept candidates passing the test in concept filtering form a set of concepts to be used in the later ontology construction.

The techniques presented in this chapter for concept extraction are simple, but sufficient for this thesis research, which focuses more on organizing concepts into hierarchies, i.e., the subtask of *relation formation*. The following two chapters describe two frameworks for *relation formation*, based on the concepts extracted by the techniques presented in this chapter.

Chapter 5

Human-Guided Ontology Learning

Besides *concept extraction*, another subtask of ontology learning is *relation formation*. *Relation formation* discovers relations among concepts and builds ontologies based on these relations. This chapter presents Human-Guided Ontology Learning [YC08a], an ontology learning framework which allows interactions between human users and machine learning algorithms during *relation formation*. Human-guided ontology learning assumes that concepts have been acquired by *concept extraction*, the techniques presented in Chapter 4, and focuses on *relation formation*.

Human-guided ontology learning is essentially a human-guided clustering framework for personal ontology learning. Through collecting manual guidance in human-computer interactions, human-guided ontology learning addresses personalization in personal ontology learning. Human-guided ontology learning is expected to produce ontologies that reflect personal preferences as a consequence of obeying manual guidance.

There are two major questions for human-guided ontology learning, and they are:

(1) Can a human-guided ontology learning approach produce ontologies with the same quality as manually-built ontologies?

(2) Can a human-guided ontology learning approach learn from individual users and capture the distinctions among their personal preferences?

To answer the above questions, this chapter consists of the following sections. Section 5.1 presents the overall human-guided ontology learning framework. Section 5.2 describes how the machine generates an initial ontology. Section 5.3 shows how to collect manual guidance from the human. Section 5.4 describes how to incorporate the manual guidance in a clustering algorithm. Section 5.5 presents the semantic feature functions used in this work. Section 5.6 briefs the simple technique used to create cluster labels. Section 5.7 studies the effectiveness of human-guided ontology learning. A user study demonstrates that human-guided ontology learning is able to generate ontologies with manually-built quality. The user study also shows that manual guidance successfully directs the machine learning process towards personal preferences.

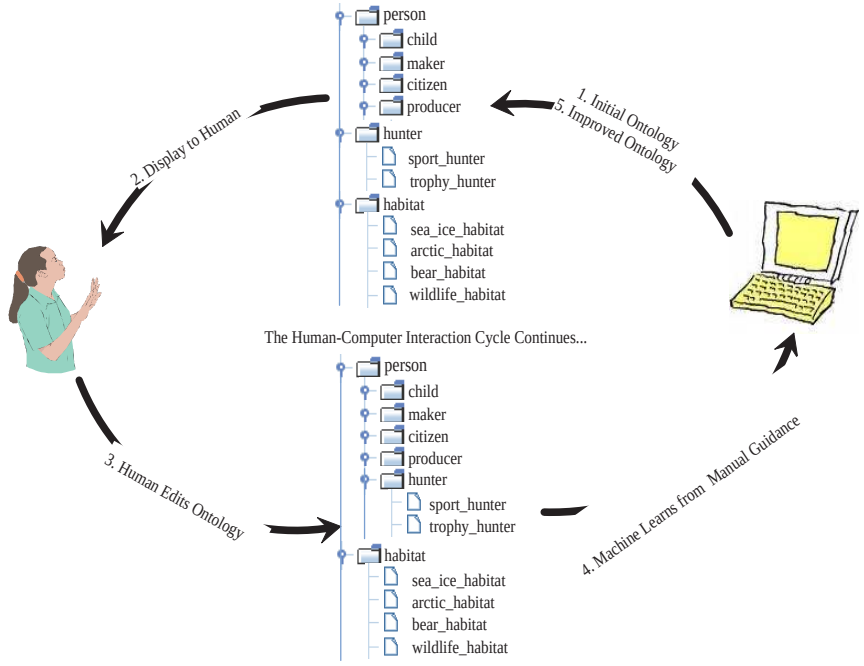


Figure 5.1: The Human-Computer Interaction Cycle.

5.1 The Human-Guided Ontology Learning Framework

Human-guided ontology learning takes a human-guided clustering approach (see Chapter 2) for personal ontology learning. Given a set of concepts, the machine first clusters the concepts and presents an initial ontology. The human then teaches the machine by providing manual guidance. The machine learns from this manual guidance and adjusts the clustering process accordingly to modify the ontology. Teaching and learning alternate until the human is satisfied.

Figure 5.1 shows the human-computer interaction cycle. In this example, the cycle starts when the machine presents an initial ontology that consists of three concept groups: *person*, *hunter* and *habitat*. The human makes a modification to the ontology by dragging and dropping the *hunter* group to be under the *person* group. This modification makes *hunter* a child concept of *person*. The machine recognizes the change, adjusts the clustering algorithm, and shows an improved ontology to the human. The human-computer interaction cycle continues until the human is satisfied with the ontology.

During each human-computer interaction, the human and the machine each modifies the ontology. Human modifications are treated as implicit guidance. The grouping of concepts before his modifications is represented by a *before matrix*; likewise, the new grouping of concepts after his modifications is represented by an *after matrix*. Manual guidance is a submatrix of the *after matrix* where there exists difference between the *before matrix* and the *after matrix*. Section 5.3 shows how to obtain manual guidance.

The machine also modifies the ontology at each human-computer interaction. The machine uses the manual guidance as training data, from which a clustering algorithm learns a distance function between concepts. The machine then applies the newly-learned distance function to obtain distance scores for ungrouped concepts in the

ontology. The clustering algorithm further clusters the ungrouped concepts based on these scores and produces a newer version of the ontology. Section 5.4 shows the details of how to incorporate manual guidance in the clustering process.

Algorithm 1 illustrates the pseudocode for human-guided ontology learning. Line 1 shows the initiation of three variables, namely U, G, and M, which are indexed by the iteration number i. U is the set of ungrouped concepts, and is initiated to be the entire set of concepts C, which is acquired by *concept extraction*. G is the set of grouped concepts, and is initiated to be empty. M is the manual guidance, which is initiated to be empty, too. Line 2 indicates the creation of the initial ontology by the machine. The initial ontology organizes the concepts into a forest that mainly consists of depth-2 or depth-3 concept hierarchies. These hierarchies are called *Ontology Fragments*. The nodes within an *ontology fragment*, except the root, are output as G. The roots of *ontology fragments* remain unorganized for the time being and are output as U. Section 5.2 presents the details of how to create the initial ontology.

Line 3-9 in Algorithm 1 show the human-computer interaction cycle. Line 4 indicates collecting manual guidance from the human. Line 5 shows that the machine learns a distance function from the manual guidance. Line 6 indicates that the machine applies this distance function to the ungrouped concepts U, in this case the roots of ontology fragments, and obtains distance scores D for these ungrouped concepts. Line 7 shows that the machine clusters the ungrouped concepts by flat clustering based on these distance scores, and updates the ontology with more grouped concepts and a new set of ungrouped concepts (the new roots).

The above process constructs an ontology in a bottom-up fashion. During each iteration, the machine clusters only one level of concepts in the ontology, and moves one level up in the next iteration. In particular, during one iteration, the machine clusters the ungrouped concepts, the roots in the hierarchy, into groups; and each newly-formed group yields one new concept, the parent concept for this group. These new parent concepts become the new roots in the hierarchy and the new set of ungrouped concepts $U^{(i+1)}$. The set of grouped concepts G grows bottom-up by including more and more new parent concepts from these iterations.

Finally, in Line 10, the algorithm outputs the latest grouped concepts G as the ontology, in which all concepts are organized based on their relations.

Since at each iteration, only one level of concepts need to be clustered, a flat clustering based only on sibling distance scores is sufficient to construct the hierarchy. The flat clustering algorithm used in this work is K-medoids [HTF01]. It randomly picks K concepts as the cluster centroids, assigns other concepts to their closest cluster centroids, recalculates the cluster centroids, and repeats assignments and recalculations until a stable cluster partition is formed. The Gap statistics [TWH00] are adopted here to estimate the optimal K, i.e., the number of clusters.

5.2 The Initial Ontology

This section presents the techniques for creating the initial ontology. These techniques includes head noun matching, WordNet hypernyms [Fel98], and lexico-syntactic patterns.

Algorithm 1: Human-Guided Ontology Learning

1. $U^{(0)}$={Ungrouped Concepts}=C, $G^{(0)}$={Grouped concepts}=$\emptyset$, $M^{(0)} = \emptyset$,$i = 1$;
2. $(G^{(1)}, U^{(1)})$ = CreateInitialOntology();
3. **while** (not Satisfied) or $U^{(i)} \neq \emptyset$
4. $M^{(i)}$=WaitforManualGuidance($G^{(i)}$,$U^{(i)}$);
5. $F^{(i)}$=LearnDistanceMetricFunction($M^{(i)}$);
6. $D^{(i)}$=PredictDistanceScores($F^{(i)}$,$U^{(i)}$);
7. $(G^{(i+1)}, U^{(i+1)})$ = ConstructClustersByFlatClustering($D^{(i)}$,$U^{(i)}$,$G^{(i)}$);
8. $i = i + 1$;
9. **end**
10. **output** $G^{(i)}$ as the Ontology.

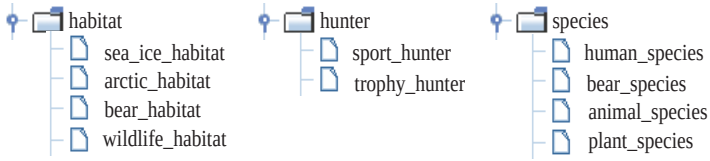


Figure 5.2: Ontology Fragments: Head Noun Matching.

5.2.1 Head Noun Matching

Head noun matching generates the initial grouping of concepts based on their head nouns. Basically, concepts that share the same head noun are grouped together. In linguistics, a head is the word that determines the syntactic type of a phrase of which this word is a member. The other elements in the phrase modify the head. For example, in *big red dog*, the word *dog* is the head, as it determines that the phrase is a noun phrase. The adjectives *big* and *red* modify this head noun. In English, noun phrases are head final, i.e., the head noun comes at the end of a phrase.

The concepts acquired by *concept extraction* include nominal unigrams, bigrams, trigrams, and named entities. Head noun matching is applied to each of these concept categories.

Head noun matching is first applied on nominal bigrams. If two or more bigrams share the same head noun, they are assigned to the same group; their common head noun is elected as the parent concept of these bigrams. For example, *pollution* becomes the parent of bigrams *water pollution* and *air pollution*. The parent concept and the child concepts form an *ontology fragment*. The above if-condition can be further relaxed: If the head nouns of two or more bigrams are from the same WordNet synset, then these bigrams are assigned to the same group; one of their head nouns is randomly selected as the parent concept. For example, *hospital* is randomly selected as the parent of bigrams *women's hospital* and *mental institution*. If a bigram does not share its head with any other bigram, it is treated as a singleton ontology fragment. Head noun matching for bigrams establishes a forest of depth-1 or depth-2 ontology fragments.

Head noun matching is then applied to nominal trigrams and named entities. Each trigram and named entity is compared with the concepts in the forest constituted by bigrams. If the suffix of a trigram or a named entity matches with an existing concept in the forest, the trigram or the named entity is added as this concept's child. For example, *heavy metal pollution* is added as *pollution*'s child. If no match is found, the trigram or the named

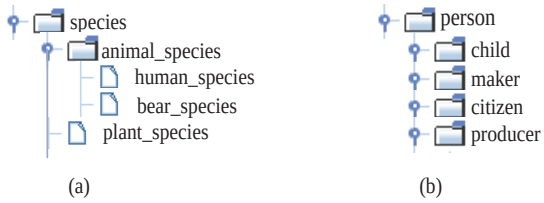


Figure 5.3: Ontology Fragments: WordNet Hypernyms.

entity is added as a singleton in the forest.

For nominal unigrams, head noun matching compares each nominal unigram with the root concepts in the forest. If a unigram is not a root concept, it is added as a singleton in the forest.

Head noun matching effectively creates the initial ontology fragments. Figure 5.2 shows examples of ontology fragments created by head noun matching.

5.2.2 Refinement Using WordNet Hypernyms

Head noun matching is simple and effective, however, it produces problematic ontology fragments. For instance, by head noun matching, *species* has two children *animal species* and *bear species*. This organization is incorrect since *animal species* should be the parent of *bear species*. Therefore, within an ontology fragment, parent and child concepts are possibly misclassified as siblings by head noun matching.

WordNet hypernyms are used to fix misclassified parent/child relations. Given two sibling concepts x and y, if x is in y's chain of hypernyms in WordNet, y is demoted as a child of x. This refinement by WordNet hypernyms creates an intermediate level within the ontology fragments. Figure 5.3(a) elaborates the refinement for *species*.

Refinement by WordNet hypernyms can also be applied among ontology fragments. As roots from different ontology fragments may have parent/child relations, the roots are examined in WordNet, and are connected as one fragment if they are in the same WordNet hypernym chain. Figure 5.3 (b) shows this refinement for *child*, *maker*, *citizen*, *producer*, and *person*; these ontology fragments are connected as one fragment since *person* is in the hypernym chains of *child*, *maker*, *citizen*, and *producer*.

5.2.3 Refinement Using Lexico-Syntactic Patterns

To further connect ontology fragments, lexical-syntactic patterns are used to identify *is-a* relations (parent/child relations) among the roots of different ontology fragments. Hearst patterns [Hea92] have been shown effective for discovering *is-a* relations. Table 5.1 gives the list of Hearst patterns used here.

Lexical-syntactic patterns are applied to the roots of ontology fragments. In particular, each pair of roots is put into the patterns and forms a text segment. For example, *toxins* and *heavy metals* are put into the pattern "NP_B and other NP_A" to form *toxins and other heavy metals*. The text segment is searched in the corpus that supplied the concepts. If the text segment is found in the corpus, the pair of roots has an *is-a* relation and is connected. Figure 5.4 shows examples of ontology fragments discovered by lexical-syntactic patterns.

Lexico-syntactic patterns refine the ontology by further connecting ontology fragments which are previously not

NP_A is the hypernym, NP_B is the hyponym.
NP_B and other NP_A
NP_B or other NP_A
NP_A such as NP_B
such NP_A as NP_B
NP_A including NP_B
NP_A, especially NP_B

Table 5.1: Hearst Patterns.

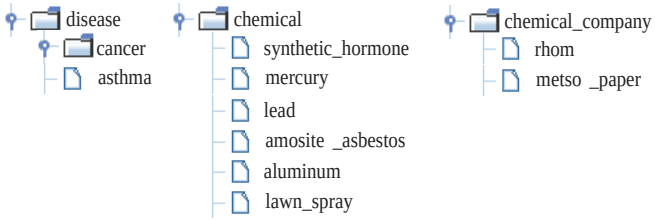


Figure 5.4: Ontology Fragments: Lexico-Syntactic Patterns.

connected by head noun matching and WordNet hypernyms. The refined ontology has more relations discovered and more fragments connected. However, the ontology remains a forest.

5.2.4 Summary

Head noun matching, WordNet hypernyms, and lexico-syntactic patterns create an initial forest of ontology fragments, which is the initial version of the ontology. The machine presents this initial ontology to the human, and waits for manual guidance.

In the initial ontology, many concepts are organized in ontology fragments; however, many others remains unorganized. The unorganized concepts are mainly the roots of ontology fragments. *Relation formation* should not stop at a forest of ontology fragments. Further organization of the concepts will be performed by *clustering*.

5.3 Collecting Manual Guidance

Manual guidance is collected during human-computer interactions via a tool called OntoCop (**Onto**logy **Co**nstruction **P**anal)[YC08a, YC08b]. OntoCop is a Java application developed in this work. Figure 5.5 shows the user interface of OntoCop. The left pane of OntoCop displays the ontology. The human can interact with the ontology by various actions, including dragging & dropping, adding, deleting, moving up/down, and promoting the concepts. The human can also edit the name of a concept. If the human wants to know more about a concept and its context, he can use the search function to retrieve the relevant documents in the corpus. The right pane of OntoCop displays the search results. The human is not required to make all modifications that he thinks are necessary; he can make only a few modifications at each human-computer interaction cycle. When the human finishes his modifications to the ontology, he triggers the machine to respond.

Human modifications can create different versions of an ontology. Each version is treated as an independent

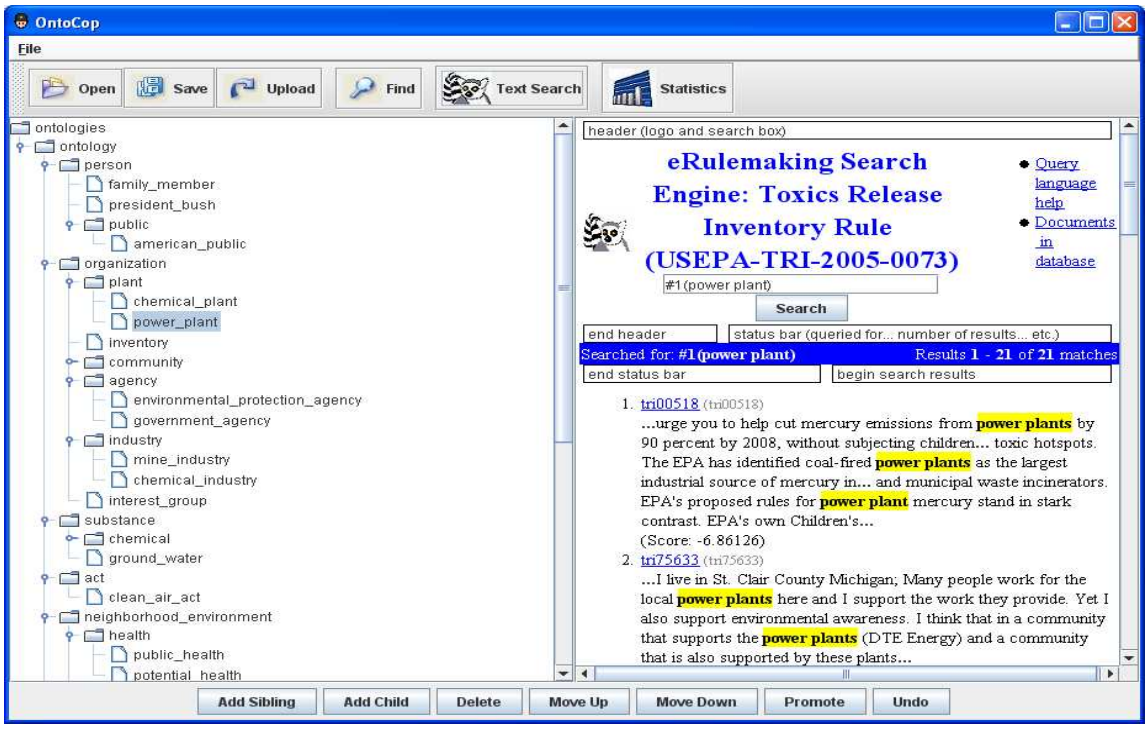


Figure 5.5: OntoCop Interface.

ontology; the fact that several versions come from the same ontology do not interfere with the algorithms presented later.

Every ontology can be represented by a matrix. Formally, an ontology with n concepts can be represented by a $n \times n$ matrix, which is called an *ontology matrix*. The $(i,j)^{th}$ entry of an *ontology matrix* indicates whether c_i, the concept at the i^{th} row, is a sibling of c_j, the concept at the j^{th} column. The value of the $(i,j)^{th}$ entry v_{ij} is defined as:

$$v_{ij} = \begin{cases} 1, & \text{if } i = j, \\ 1, & \text{if } c_i \text{ is a sibling of } c_j,\ i \neq j, \\ 0, & \text{if } c_i \text{ is not a sibling of } c_j,\ i \neq j. \end{cases} \tag{5.1}$$

For example, in Figure 5.6, the example ontology before human modifications can be represented as a *before matrix*:

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Figure 5.6: An Example Ontology Before and After Human Modifications (Concept Set Unchanged).

where concept set is $\{a, b, c, d, e\}$, the rows and the columns are ordered alphabetically for the concepts. Likewise, the example ontology after human modifications can be represented as an *after matrix*:

$$B = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

The manual guidance M is a submatrix which consists of some entries of the *after matrix* B; at these entries, there exists difference between the *before matrix* A and the *after matrix* B. Formally,

$$M = B[r; c]$$

where $r = \{i : b_{ij} - a_{ij} \neq 0\}$, $c = \{j : b_{ij} - a_{ij} \neq 0\}$, a_{ij} is the $(i, j)^{th}$ entry in A, and b_{ij} is the $(i, j)^{th}$ entry in B.

For the example in Figure 5.6, the difference between B and A is:

$$B - A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

The 2^{nd}, 3^{rd} and 4^{th} rows of $(B - A)$ and the 2^{nd}, 3^{rd} and 4^{th} columns of $(B - A)$ contain non-zero entries, which indicate existence of difference between A and B. Hence, the manual guidance M is:

$$M = B[2, 3, 4; 2, 3, 4] = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} = \begin{array}{c|ccc} & b & c & d \\ b & 1 & 0 & 0 \\ c & 0 & 1 & 1 \\ d & 0 & 1 & 1 \end{array}.$$

This simple example illustrates the case when the set of concepts remain unchanged before and after human

Figure 5.7: An Example Ontology Before and After Human Modifications (Concept Set Changes).

modifications. Usually, a series of human modifications, which only involve dragging and dropping, moving up/down, and promoting concepts, yield unchanged concept set. However, often the human adds, deletes or renames concepts, hence the concept set changes. When the concept set changes, the above definition of manual guidance M needs a slight alteration.

Figure 5.7 shows an example ontology whose concept set changes: concept b is deleted from the ontology; concept f is added as a new concept in the ontology. Therefore the concept set changes from $\{a, b, c, d, e\}$ to $\{a, c, d, e, f\}$. The *before matrix* A is the same as in the previous example; the *after matrix* B becomes:

$$B = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Here we define the set of concepts in the ontology before modifications as C_A, and the set of concepts in the ontology after modifications as C_B. We also define the expanded set of concepts, C_E, as the union of C_A and C_B:

$$C_E = C_A \bigcup C_B = C_A \bigcup \{c_k : c_k \in C_B, c_k \notin C_A\} = C_B \bigcup \{c_k : c_k \in C_A, c_k \notin C_B\}.$$

The *expanded before matrix* A' and the *expanded after matrix* B' are both defined over C_E. The expanded rows and columns in A' and B' are filled with 0 for non-diagonal entries, and 1 for diagonal entries. For the example in Figure 5.7, the *expanded before matrix* A' is:

$$A' = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Note that the expanded set of concepts C_E is $\{a, b, c, d, e, f\}$. The 6^{th} row and 6^{th} column are newly expanded.

They correspond to concept f, which is added to the ontology.

The *expanded after matrix* B' is:

$$B' = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Note that the 2^{nd} row and 2^{nd} column are newly expanded. They correspond to concept b, which is deleted from the ontology.

The manual guidance M is a submatrix which consists of some entries of the *after matrix* B; at these entries, there exists difference from the *expanded before matrix* A' to the *expanded after matrix* B'. Note that the concepts corresponding to these entries should exist in C_B, the unexpanded set of concepts after human modifications. Formally,

$$M = B[r; c]$$

where $r = \{i : b_{ij} - a_{ij} \neq 0,\ c_i \in C_B\}$, $c = \{j : b_{ij} - a_{ij} \neq 0,\ c_j \in C_B\}$, a_{ij} is the $(i,j)^{th}$ entry in A', and b_{ij} is the $(i,j)^{th}$ entry in B'.

For the example in Figure 5.7, the difference between B' and A' is:

$$B' - A' = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

The 2^{nd} to 6^{th} rows of $(B' - A')$ and the 2^{nd} to 6^{th} columns of $(B' - A')$ contain non-zero entries, which indicate existence of difference between A' and B'. Among the rows and columns, only the 3^{rd} to 6^{th} rows, and the 3^{rd} to 6^{th} columns exist in the original *after matrix* B; and these rows and columns correspond to the 2^{nd} to 5^{th} rows, and the 2^{nd} to 5^{th} columns of B. Hence, the manual guidance M is:

$$M = B[2,3,4,5;2,3,4,5] = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} = \begin{array}{c|cccc} & c & d & e & f \\ c & 0 & 0 & 0 & 1 \\ d & 0 & 0 & 1 & 0 \\ e & 0 & 1 & 0 & 0 \\ f & 1 & 0 & 0 & 0 \end{array}.$$

Manual guidance, which is collected from the ontology before and after human modifications, indicates the

	sea_ice_habitat	arctic_habitat	bear_habitat	wildlife_habitat	child	maker	citizen	producer	hunter	wolf_territory	bear_territory
sea_ice_habitat	0	0	0	0	1	1	1	1	1	0	0
arctic_habitat	0	0	0	0	1	1	1	1	1	0	0
bear_habitat	0	0	0	0	1	1	1	1	1	0	0
wildlife_habitat	0	0	0	0	1	1	1	1	1	0	0
child	1	1	1	1	0	0	0	0	0	0	0
maker	1	1	1	1	0	0	0	0	0	0	0
citizen	1	1	1	1	0	0	0	0	0	0	0
producer	1	1	1	1	0	0	0	0	0	0	0
hunter	1	1	1	1	0	0	0	0	0	0	0
wolf_territory	0	0	0	0	0	0	0	0	0	1	1
bear_territory	0	0	0	0	0	0	0	0	0	1	1

Figure 5.8: Training Distance Matrix.

human's preference of how to construct the ontology. The concepts in manual guidance are the focus which captures the human's attention. The positive entries in manual guidance indicate the human's preference of grouping the corresponding concepts together; the non-positive entries indicate his preference of keeping the corresponding concepts apart. This grouping information in manual guidance, which may be biased towards the human's personal preference, will be used to cluster other concepts in the ontology.

5.4 Incorporating Manual Guidance

In the human-guided ontology learning framework shown in Algorithm 1, there are three major variables, the ungrouped concepts U, the grouped concepts G, and the manual guidance M. Section 5.2 presented how to get the initial U and G. Section 5.3 presented how to collect M from the human. This section presents a supervised distance learning and predicting algorithm to incorporate manual guidance in the clustering process.

During each human-computer interaction, the machine learns a distance function from concepts in M or G, and further organizes concepts in U based on this distance function. At each iteration, the machine updates U and G. In particular, at the i^{th} iteration of human-computer interactions, $U^{(i)}$, $G^{(i)}$, and $M^{(i)}$ denote the current ungrouped concepts, grouped concepts and manual guidance, respectively.

Manual guidance is used as the training data for the supervised distance learning algorithm. In particular, the training data consists a set of concepts $\mathbf{x}^{(i)}$ and its corresponding distance matrix $\mathbf{y}^{(i)}$. Each set $\mathbf{x}^{(i)}$ represents a set of concepts at the human-computer interaction indexed by i. In particular, $\mathbf{x}^{(i)}$ contains the concepts in $M^{(i)}$. The corresponding distance matrix $\mathbf{y}^{(i)}$ gives the the correct partition (clustering) for $\mathbf{x}^{(i)}$. The entry of $\mathbf{y}^{(i)}$ which corresponds to concept $x_j^{(i)}$ and $x_k^{(i)}$ is $y_{jk}^{(i)} \in \{1, 0\}$, where $y_{jk}^{(i)} = 0$, if $x_j^{(i)}$ and $x_k^{(i)}$ are in the same cluster; 1, otherwise. Note that $\mathbf{y}^{(i)} = 1 - M^{(i)}$.

Figure 5.8 shows a training distance matrix $\mathbf{y}^{(i)}$. The training distance matrix elaborates the distance between *child*, *maker*, *citizen*, *producer*, *hunter*, *sea ice habitat*, *arctic habitat*, *bear habitat* and *wildlife habitat*. In the training distance matrix, within-cluster distance is defined as 0 and between-cluster distance is defined as 1.

The goal of supervised distance learning is to learn a good pairwise distance metric function which best preserves the regularity in the training distance matrix. In this work, the estimated pairwise distance metric

function is represented as a Mahalanobis distance function [Mah36].

$$d_{jk} = \sqrt{(x_j - x_k)^T S^{-1} (x_j - x_k)} \tag{5.2}$$

Mahalanobis distance measures the correlation between two concepts by assigning adaptive weights to different underlying feature functions. The feature functions can be any function which measures semantic relations between two concepts. For example, the feature functions can include contextual features, co-occurrence, syntactic dependency, lexical-syntactic patterns, word length difference and definition overlap. These heterogonous feature functions evaluate the semantic relation between two concepts from different aspects and aim to capture a wide range of characteristics of semantic relations. Section 5.5 will give more details about the feature functions.

One important characteristic of a distance matrix is that it must represent valid clustering partitions, which means the clustering partitions represented by the distance matrix should be consistent. Therefore, certain constraints need to be satisfied. An obvious one is that concepts in the same cluster should have smaller distance scores than those in different clusters. Moreover, a valid distance metric should be non-negative and satisfy the triangle inequality. To ensure such regularities, we need to constrain the parameter matrix S to be positive semi-definite (PSD) [Bha06].

Theoretically, the parameter estimation problem in our settings is to get S such that the expected loss is minimized. The expected loss, or the risk, can be represented as :

$$R(\hat{\mathbf{y}}) = \int \Delta(\mathbf{y}, \hat{\mathbf{y}}) p(\mathbf{x}, \mathbf{y}) d\mathbf{x} d\mathbf{y} \tag{5.3}$$

where $p(\mathbf{x}, \mathbf{y})$ is the joint distribution of concepts and their clustering assignments. The loss function Δ in Equation 5.3 is defined as the squared error loss of two distance matrices.

The expected loss is minimized through minimizing the squared errors. The optimization function is then defined as :

$$\min_S \sum_{j=1}^{|\mathbf{X}^{(i)}|} \sum_{k=1}^{|\mathbf{X}^{(i)}|} \left(y_{jk}^{(i)} - \sqrt{\Phi(x_j^{(i)}, x_k^{(i)})^T S^{-1} \Phi(x_j^{(i)}, x_k^{(i)})} \right)^2 \tag{5.4}$$

$$\text{subject to } S \succeq 0$$

where $\Phi(x_j^{(i)}, x_k^{(i)})$ represents a set of pairwise underlying feature functions, where each feature function is $\phi_d : (x_j^{(i)}, x_k^{(i)}) \mapsto r \in \Re$ with d=1,...,$|\Phi|$. The underlying feature functions evaluates the relation between $(x_j^{(i)}, x_k^{(i)})$ from various aspects. S is a parameter matrix, which weighs the underlying distance feature functions.

The optimization can be done by any standard semi-definite programming (SDP) solvers. Matlab software packages sedumi[1] and yalmip[2] are used to perform the optimization.

Given the learned parameter matrix S, it is easy to generate distance scores for any pair of ungrouped concepts in U. Let $(x_l^{(i+1)}, x_m^{(i+1)})$ be an ungrouped concept pair from $U^{(i)}$. By calculating the distance for each concept pairs, we obtain the entries in a new distance matrix $\hat{\mathbf{y}}^{(i+1)}$. Note that this distance matrix should also result in a consistent clustering, which is guaranteed by the positive semi-definiteness of the parameter matrix S. The

[1] http://sedumi.mcmaster.ca/

[2] http://control.ee.ethz.ch/~joloef/yalmip.php

entry values for $\hat{\mathbf{y}}^{(i+1)}$ is defined as:

$$\hat{y}_{lm}^{(i+1)} = \sqrt{\Phi(x_l^{(i+1)}, x_m^{(i+1)})^T S^{-1} \Phi(x_l^{(i+1)}, x_m^{(i+1)})} \tag{5.5}$$

The learned distance matrix $\hat{\mathbf{y}}^{(i+1)}$ contains the distance scores for concepts in $U^{(i)}$. Note that previously they were unorganized. The K-medoids clustering algorithm further clusters the ungrouped concepts based on their scores and produces the next version of the ontology.

The newly grouped concepts are added into $G^{(i)}$ and form the new set of grouped concepts $G^{(i+1)}$. The grouped concepts forms a forest of ontology fragments. Each new group is represented by a new concept, which is labeled by a simple concept labeling algorithm described later in Section 5.6. These new concepts become new roots of the ontology, and form the new set of ungrouped concepts $U^{(i+1)}$. The algorithm updates $G^{(i+1)}$ and $U^{(i+1)}$ and goes into the next iteration in a bottom-up fashion. OntoCop then presents the modified ontology to the human and waits for the next round of manual guidance.

In summary, in human-guided ontology learning, the machine requests for manual guidance at each iteration, and adjusts the the distance metric function accordingly. In particular, by taking into account the human's modification to the ontology, the machine learns from his/her personalized grouping of concepts. The training data is updated at each learning cycle and feed into the distance learning algorithm, which allows the formation of an ontology to be based on the personal preferences from individuals.

5.5 The Features

The features used in this work are indicators of semantic relations between concepts [YC09a]. Given two concepts c_x, c_y, a feature is defined as a function generating a single numeric score $h(c_x, c_y) \in \Re$ or a vector of numeric scores $h(c_x, c_y) \in \Re^n$. The feature types include *contextual*, *co-occurrence*, *syntactic dependency*, *lexical-syntactic patterns*, and *miscellaneous*.

The first set of features captures *contextual* information of concepts. According to the Distributional Hypothesis [Har54], words appearing in similar contexts tend to be similar. Therefore, word meanings can be inferred from and represented by contexts. Based on the hypothesis, we develop the following contextual features.

Global Context KL-Divergence: The global context of each concept is the search results collected through querying search engines against several corpora (see the auxiliary datasets in Chapter 3). It is built into a language model for each concept. This feature function measures the Kullback-Leibler divergence (KL divergence) between the language models associated with the two inputs.

Local Context KL-Divergence: The local context is the collection of all the left two and the right two words surrounding a concept. Similarly, the local context is built into a language model for each concept; the feature function outputs KL divergence between the models.

The second set of features is *co-occurrence*. In this work, co-occurrence is measured by point-wise mutual information between two concepts:

$$pmi(c_x, c_y) = log \frac{Count(c_x, c_y)}{Count(c_x)Count(c_y)} \tag{5.6}$$

Hypernym Patterns	*Sibling Patterns*
NP_x (,)?and/or other NP_y	NP_x and/or NP_y
such NP_y as NP_x	*Part-of Patterns*
NP_y (,)? such as NP_x	NP_x of NP_y
NP_y (,)? including NP_x	NP_y's NP_x
NP_y (,)? especially NP_x	NP_y has/had/have NP_x
NP_y like NP_x	NP_y is made (up)? of NP_x
NP_y called NP_x	NP_y comprises (of)? NP_x
NP_x is a/an NP_y	NP_y consists of NP_x
NP_x , a/an NP_y	

Figure 5.9: List of Lexico-Syntactic Patterns.

Based on different definitions of $Count(.)$, we have the following co-occurrence features.

Document PMI: This feature function measures document-level PMI, where $Count(.)$ is defined as the number of documents in the corpus containing the concept(s).

Sentence PMI: This feature function measures sentence-level PMI, where $Count(.)$ is defined as the number of sentences in the corpus containing the concept(s).

Google PMI: This feature function measures web-based PMI, where $Count(.)$ is defined as n as in "Results 1-10 of about n for *concept*" appearing on the first page of Google search results for a concept or the concatenation of a concept pair.

The third set of features employs *syntactic dependency* analysis. We use Assert [3] to label the semantic roles.We have the following syntactic dependency features.

Minipar Syntactic Distance: This feature function measures the average length of the shortest syntactic paths (in the first syntactic parse tree returned by Minipar[4]) between two concepts in sentences containing them.

Modifier Overlap: This feature function measures the number of overlaps between two concept's modifiers for the two concepts in sentences containing them.

Object Overlap: This feature function measures the number of overlaps between objects in sentences with the two concepts as subjects.

Subject Overlap: This feature function measures the number of overlaps between subjects in sentences with the two concepts as objects.

Verb Overlap: This feature function measures the number of overlaps between verbs for the two concepts in sentences containing them.

The fourth set of features is *lexical-syntactic patterns*. Table 5.9 lists all patterns used in this work. The features include:

Hypernym Patterns: This feature function is based on patterns proposed by (Hearst, 1992) [Hea92] and (Snow et al., 2005) [SJN05]. The feature function returns a vector of scores for two concepts, one score per pattern. A score is 1 if two concepts match a pattern in text, 0 otherwise.

[3] http://cemantix.org/assert.
[4] http://www.cs.ualberta.ca/lindek/minipar.htm.

Sibling Patterns: This feature function contains basically conjunction patterns. The feature function returns a vector of scores for two concepts, one score per pattern. A score is 1 if two concepts match a pattern in text, 0 otherwise.

Part-of Patterns: This feature function is based on patterns proposed by (Girju et al., 2003) [GBM03] and (Cimiano and Wenderoth, 2007) [CW07]. The feature function returns a vector of scores for two concepts, one score per pattern. A score is 1 if two concepts match a pattern in text, 0 otherwise.

The last set of features is *miscellaneous*, including:

Word Length Difference: This feature function measures the length difference between two concepts.

Definition Overlap: This feature function measures the number of word overlaps between the definitions obtained by querying Google with "define:*concept*".

These heterogeneous features vary from simple statistics to complicated syntactic dependency features, basic word length to comprehensive Web-based contextual features. The flexible design of our learning framework allows us to use all of them.

5.6 Concept Labeling

Semantic labeling for clusters is a challenging problem for the clustering-based approaches. Caraballo (1999) used Hearst's patterns to find hypernym candidates for each class members and then select the top candidate related to the largest number of class members [Car99]. Pantel and Ravichiandran (2004) used four lexical-syntactic patterns to identify and use mutual information to rank hypernym candidates related to a group of core class members, which they called "committee" [PR04].

In the proposed human-guided bottom-up clustering procedure, when K-medoids algorithm creates a new group for the top level concepts, the new group is nameless, i.e., without a parent concept. It needs to assign meaningful names to these newly-created parent nodes. The name for the new node should be general enough to cover the scope of all the child concepts, and be specific enough to just cover that of them. We use the Web as the provider of general knowledge. A query formed by concatenating the child concepts is sent to the Google search engine. The top 10 returned snippets is parsed. After stopwords removal, the most frequent phrase is selected as the name for the new parent. For example, *president* becomes the parent of concepts *Bush* and *Reagan*.

5.7 User Study and Experiments

To evaluate the system performance and answer the two questions posed at the beginning of the chapter, a user study was conducted. The task is defined in the domain of public comments, where administrative agencies of the U.S. government seek comments from stakeholders and the public about proposed regulations. The situation given in the evaluation is that the agencies need to organize the comments into rule-specific ontologies based on their short-term needs. Public comment datasets are typically used only for 1-6 months, during the period in which the final rule is drafted.

We collaborated with the Qualitative Data Analysis Program (QDAP) at the University of Pittsburgh's University Center for Social and Urban Research (UCSUR) to conduct the user evaluation. Twelve professional

Table 5.2: Intercoder Agreements (Kappa) on Parent-Child Pairs.

	Manual-Manual	Manual-Interactive	t	p
Wolf	0.55	0.55	0	0.50
Polar Bear	0.44	0.46	0.21	0.42
Mercury	0.61	0.51	1.89	0.03

Table 5.3: Intercoder Agreements (Kappa) on Sibling Pairs.

	Manual-Manual	Manual-Interactive	t	p
Wolf	0.43	0.42	0.16	0.56
Polar Bear	0.36	0.38	0.21	0.42
Mercury	0.43	0.23	3.77	<0.001

coders familiar with the problem domain participated in the experiment. They were divided into two groups: four for the manual group and eight for the interactive group. Users in the manual group were asked to construct an ontology with the concept candidates produced by the system in a bottom-up fashion until they felt satisfied with their work or reached a 90-minute limit. The interactive group were asked to work interactively with the system until they felt satisfied with the work or reached a 90-minute limit. Each user in the interactive group worked on organizing the concept candidates for a few minutes, then uploaded the modified hierarchy to the system; the system learned from user feedback, produced a new hierarchy and returned it to the user. It is a user's decision to continue modifying the ontology and teaching the system to learn or stop. Both groups used the same editing tool provided in OntoCop, such as deleting, adding a node, dragging and dropping a node, promoting a node to the higher level, undoing previous actions, etc. The set of concept candidates given to both groups were the same.

For the four public comment datasets (see Chapter 3) used in the experiments, "TRI" is the one with the smallest vocabulary. It is used to train both manual and interactive users. The experimental results generated on "Wolf", "Polar Bear" and "Mercury" datasets are reported in the following sections.

For a given ontology, two lists of concept pairs are generated. The first is a list of all parent-child pairs in the hierarchy. The second is a list of all sibling pairs. Performance metrics for parent-child pairs measure whether a concept is assigned to the correct parent while that for sibling pairs measure whether a concept belongs to the correct group.

5.7.1 Quality of Interactively vs. Manually Constructed Ontologies

This experiment investigates whether the proposed guided machine learning approach is able to produce ontologies with the same quality as manually built ones. We compare the intercoder agreement between two manual runs and that between one manual and one interactive run in this experiment. Cohen's Kappa statistic [Coh60] is used to assess intercoder agreement, which is defined as:

$$\kappa = \frac{p(A) - p(E)}{1 - p(E)}$$

Table 5.4: Average Manual Editing Costs.

	add	delete	move	name change	undo	total
manual	57	200	2807	71	19	3153
interactive	21	129	1694	40	8	1890

where $p(A)$ is the observed agreement between the assessments from two coders, which can be calculated as $(a+d)/m$, where a is the number of positive pairs in both assessments, b is the number of pairs which are positive in assessment one but negative in assessment two, c is the number of pairs which is negative in assessment one but positive in assessment two, d is the number of negative pairs in both assessments, $m = a+b+c+d$. $p(E)$ is the agreement expected by chance, which can be calculated as $(a+b)(a+c)/m^2+(b+c)(c+d)/m^2$. In computational linguistics $\kappa > 0.67$ has often been required to draw any conclusions of agreement. However, there has been a number of objections to this standard and less strict evaluations consider $0.2 < \kappa < 0.4$ to indicate fair agreement and $0.4 < \kappa < 0.6$ to indicate moderate agreement [Car96].

In this set of experiments, the intercoder agreement measured by Kappa between two manual runs is averaged over 4x3=12 pairs of manual-manual runs. The intercoder agreement between manual and interactive runs is averaged over 4x8=32 pairs of manual-interactive runs. Table 5.2 and 5.3 show the averaged intercoder agreements and the significance test results for parent-child pairs and sibling pairs respectively.

From Table 5.2 we can see that both the intercoder agreement between manually built ontologies and that between manual-interactive runs are within the range of 0.44 to 0.61, which indicates moderate agreement. This can be attributed to the fact that ontology construction is a highly subjective task where each user tends to produce different hierarchy to reflect the knowledge structure in their own minds. Another factor could be that the users are given lots of concepts within a relatively short time and they are not familiar with the proposed rules. However, we also observe that manual-interactive intercoder agreement is comparable with manual-manual intercoder agreement, which indicates that the guided machine learning approach is able to produce the same quality ontologies as humans do. Table 5.3 demonstrates the same conclusion for sibling pairs.

A series of one-tailed t-tests also confirm it. Almost all significant test results are not significant, $t < 2$ and $p > 0.01$, which show no statistical significant differences between pairs of manually-built ontologies and interactively-built ontologies. The only exception is for the sibling pairs in the mercury dataset, where a significant difference is observed ($p < 0.001$). Further investigation shows that within the 90-minute time limit, manual users are not able to finish the ontology construction task for the mercury dataset, and leave 105 top level concepts ungrouped in average as reported in Table 5.6, while interactive users are able to further group the top level concepts into 85 groups which is much less than what manual users do. The different degrees of completeness result in differences in the intercoder agreements. Nevertheless, the results demonstrate that guided machine learning is able to produce the same quality ontologies as humans do.[5]

5.7.2 Costs of Interactively vs. Manually Constructed Ontologies

This experiment investigates the construction costs of taking manual or interactive approach. We compare the construction logs for users from both manual and interactive groups. Table 5.4 shows the number of manual

[5]See the Appendix for three random selected ontologies created by manual and interactive runs.

Table 5.5: Ontology Construction Duration.

	Wolf	Polar Bear	Mercury	**average**
manual	1:24	1:22	1:33	**1:27**
interactive -human	0:33	0:29	0:30	**0:31**
-computer	0:33	0:05	0:35	**0:24**

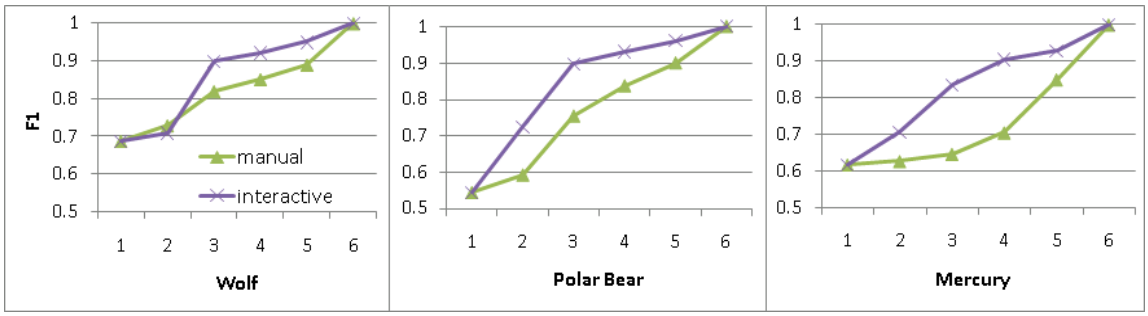


Figure 5.10: Learning Effect for Parent-Child Pairs over Cycles.

editings of building ontologies for three datasets. The editings include adding a (child or sibling) concept, moving a concept by drag & drop, deleting a concept, changing name for a concept and undoing previous actions. In total, interactive users use 40% less editing actions to produce the same quality ontologies as manual users do. A one-tailed t-test shows a significant reduction, t=10 and $p < 0.001$, of interactive runs in editing costs as compared to manual runs. It demonstrates that guided machine learning is significantly more cost effective than manual work.

We also compare the ontology construction duration. Table 5.5 shows the actual time needed to construct an ontology for both manual and interactive runs. It also shows the amount of time spent by human users in the interactive runs in the brackets. In general, interactive runs save 30 to 60 minutes for building one ontology. Within an interactive run, a human user only needs to spend 31 minutes in average to construct an ontology, which is 64% less than 1 hour and 27 minutes in a manual run. It shows that guided machine learning greatly saves a human user's time to construct an ontology.

5.7.3 Learning from Personal Preferences

In human-guided ontology learning, human users and the system work together to build an ontology in several learning cycles. This experiment investigates the system's ability to learn from personal preferences from different users and eventually fulfil their personal needs.

This experiment studies the ontology quality before and after each learning cycle for both interactive and manual users. The gold standard ontology for each user is his/her own finalized ontology. At each learning cycle, a user's current, partial ontology is compared with his/her own gold standard in terms of F1-measures. F1-measure is calculated as 2*P*R/(P+R), where P is Precision and is calculated as the number of correctly returned parent-child (or sibling) pairs divided by the number of returned parent-child (or sibling) pairs, R is Recall and is calculated as the number of correctly returned parent-child (or sibling) pairs divided by the total

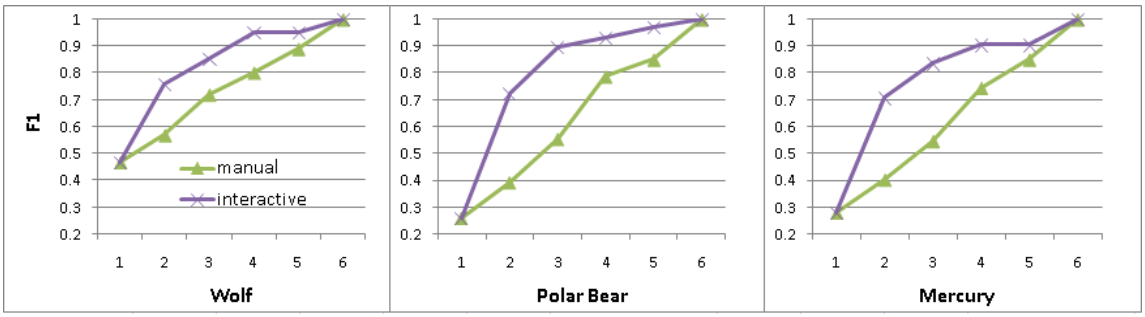


Figure 5.11: Learning Effect for Sibling Pairs over Cycles.

number of correct parent-child (or sibling) pairs in the ground truth.

For interactive users, ontologies that uploaded by them at each learning cycle are evaluated based on the corresponding ground truth. The reported F1-measure for the interactive group is averaged over the 8 interactive users. For manual users, their partially constructed ontologies with 20%, 40%, 60%, and 80% modifications in their editing logs are evaluated based on the corresponding ground truth. These partial ontologies based on modification percentage in the manual runs are comparable to the partial ontologies created at each learning cycle in the interactive runs. The reported F1-measure is averaged over the 4 manual users.

Figure 5.10 shows the learning curves for both manual and interactive runs based on the parent-child pairs over six learning cycles. Figure 5.11 shows the learning curve for both manual and interactive runs based on the sibling pairs over six learning cycles. The x-axis are the learning cycles for each dataset. The y-axis indicates the averaged F1-measures.

In both Figure 5.10 and 5.11, F1-measures for both manual and interactive groups converge to 1 at the end of the learning process. It is because that this is a personalized task and each individual's finalized ontology is used as the gold standard. For interactive users, we notice an obvious performance gain between an uploaded ontology and the ontology learned automatically from it. It shows that the interactive system learns from periodic manual guidance and improves the F1-measure at each learning cycle. Moreover, comparing the F1-measure of interactive and manual users, we notice that the learning curve of the interactive users are steeper than that of the manual users. It indicates that the human guided ontology learning not only learns from personal preferences but also helps the interactive users move faster towards their personal satisfaction levels.

5.7.4 Structure Changes over Learning Cycles

This experiment studies the changes of ontology structures over the learning cycles. Table 5.6 lists the total number of concepts, hierarchy depth, number of concepts at level 1 (the top level) and level 2, averaged over 8 interactive users for the three datasets in six learning cycles. The structure of the manually constructed ontologies is also presented and averaged over 4 manual users.

From Table 5.6, we notice that with more learning cycles, the depth of the ontology increases, number of top level nodes decreases. Comparing to the manual runs, interactive runs generally produce deeper hierarchies and fewer top level nodes. This indicates that interactive users have developed more ontology layers and created more

Table 5.6: Ontology Structure over Learning Cycles: Averaged over 12 Users.

	wolf				polar bear				mercury			
cycle	#nodes	depth	level1	level2	#nodes	depth	level1	level2	#nodes	depth	level1	level2
manual	766	5.8	13.8	74.5	335	5.5	7	45.8	1084	4.3	104.8	186.8
1	795	4	103	235	351	4	87	102	1046	3	240	296
2	766	4.3	85.7	203	324	4.3	70.3	87.3	1032	4.2	190.2	239.2
3	611	4.8	23.2	96.8	318	4.7	30.8	69	1006	4.5	158	234.7
4	694	5.7	10.7	75	331	4.8	20	61	938	4.8	35	159.4
5	648	7	10	49	314	5	8.5	52.5	918	4	53.5	170.5
6	-	-	-	-	-	-	-	-	868	5	84	217

groupings thus produced fewer top level nodes than manual users. It also suggests that our interface design and learning algorithm create more opportunities for a user to explore the structure of an ontology, which may end up with an ontology with better quality within a fixed time frame.

5.8 Summary

This chapter proposed a human-guided learning framework to construct personal ontology given a text corpus. By incorporating personal preferences as guidance, the proposed distance metric learning and supervised hierarchical clustering framework is able to predict good semantic distance scores for concepts and further organize them into ontology hierarchies.

A user study on concept ontology construction over large email datasets has been done. The results show that the interactive learning process not only saves time and human efforts, but also produces high quality ontology by learning from individual users and accelerates the process of developing personal ontologies.

Human-guided ontology learning has been tested on datasets which do not contain broad, diverse concepts like many other research have been worked on, which makes the ontology construction a harder problem. Appendix A shows examples of ontologies for the TRI dataset. Such specific domains contain many similar concepts and the task requires stronger ability to disambiguate the subtle differences among similar concepts. Human-guided ontology learning has successfully demonstrated its ability to deal with such domains.

Chapter 6

Metric-Based Ontology Learning

This chapter presents the metric-based ontology learning framework [YC09b]. This framework incrementally clusters concepts based on *ontology metric*, a score indicating the semantic distance between concepts; and transforms the task of ontology learning into a multi-criterion optimization based on minimization of ontology structures and modeling of concept abstractness.

Metric-based ontology learning flexibly incorporates a wide range of semantic features for ontology learning. Similar to human-guided ontology learning (Chapter 5), metric-based ontology learning uses heterogenous features ranging from simple statistics to complicated syntactic dependency features. Moreover, metric-based ontology learning offers a more general framework which allows a further study on which features are the best for ontology learning at different concept abstraction levels and for ontology learning of different relations.

Metric-based ontology learning addresses concept abstractness in ontologies. Unlike most prior research (see Chapter 2), including Chapter 5, which uses a single rule or a single feature function to infer the semantic relations between concepts at all levels of concept abstraction, metric-based ontology learning supports different feature functions at different abstraction levels. Strictly speaking, it isn't required that metric-based ontology learning adopt a one-at-a-time approach. This more general framework has the potential to learn more complex ontologies than previous approaches.

In addition, metric-based ontology learning takes an incremental clustering approach to organize the concepts into ontologies. The learning framework builds an ontology step by step by considering the concepts one by one and placing each concept at the optimal position in the concept hierarchy. Most ontology learning algorithms [Blo05, CV05b, CSV04, Hea92, KAB04, KW02, SWGM05, SC99, SJN06, YC08b], including the human-guided ontology learning framework (Chapter 5), cluster concepts agglomeratively by merging similar concepts together, labeling new clusters, and repeating the process. On the contrary, metric-based ontology learning avoids cluster labeling, the most difficult step in these approaches, and focuses on finding optimal position for each concept by inserting each concept into the ontology in a way that minimizes the changes to ontology structures and models concept abstractness.

This chapter includes the following sections. Section 6.1 introduces terminologies used in the metric-based ontology learning framework. Section 6.2 states two important assumptions used in this framework: minimum evolution assumption and abstractness assumption. Section 6.3 presents the metric-based ontology learning

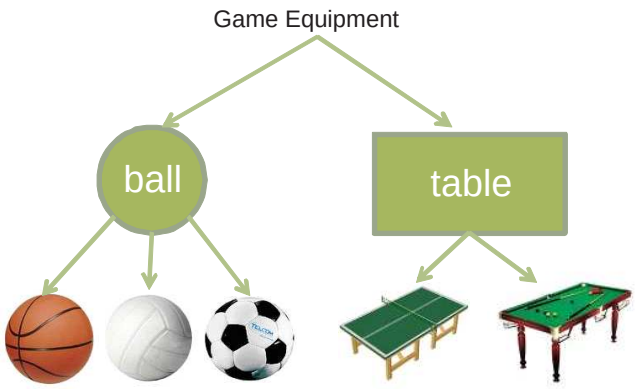


Figure 6.1: A Full Ontology.

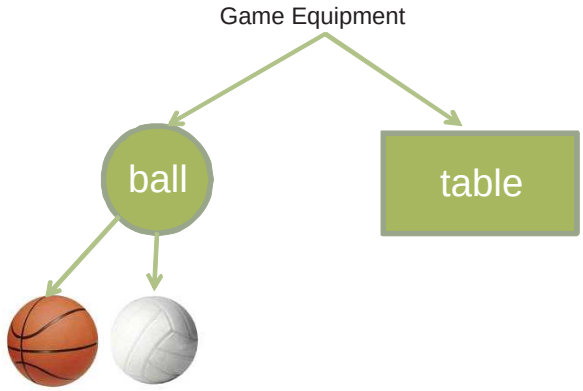


Figure 6.2: A Partial Ontology.

framework. Section 6.4 shows the evaluation and experiments, which not only show that our system achieves higher F1-measure than three state-of-the-art systems, but also reveals the interaction between features and various types of relations, as well as the interaction between features and concept abstractness.

6.1 Ontologies, Ontology Metric, and Information Functions

To have a theoretical formulation of the metric-based ontology learning framework, this section presents definitions of ontologies and related terminologies.

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

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

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

A *full ontology* $\mathtt{S(T:T} \in full)$ is a tree containing all the concepts in C. A *partial ontology* $\mathtt{S(T:T} \notin full)$ is a tree containing only a subset of concepts in C. Figure 6.1 shows an example of a full ontology. Figure 6.2 shows one example of a partial ontology for the particular full ontology shown in Figure 6.1.

Ontology Metric 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 $\mathtt{T}$. Here we assume that the graph structure of an ontology is a tree. A tree metric is defined as a function that applies only to the set of leaf nodes in the tree. Although an ontology metric is similar to *tree metric* used in graph theory [Tut01], ontology metric is especially designed for the metric-based ontology learning framework and is a function which can be applied on both leaf and non-leaf nodes; tree metric in graph theory can only be applied to leaf nodes.

The ontology metric d on an ontology $\mathtt{T}$ with edge weights w for any data pair (i, j) in the concept set $\mathtt{C}$ is

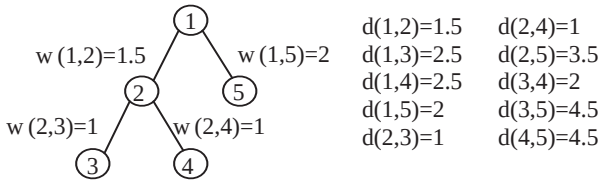


Figure 6.3: Illustration of Ontology Metric.

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 6.3 gives an illustration of ontology metric on a 5-node ontology. As a valid metric, an ontology metric 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. $d(i,j) + d(j,k) \geq d(i,k)$

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 and non-overlapped paths equals the ontology metric for the ending points on the entire path:

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

Moreover, 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* [Tut01]: for any four concepts (i,j,s,t) in $\mathtt{T}$,

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

Information Functions The amount of information in an ontology T is measured and represented by an information function $Info(T)$. An information function is defined as the sum of the ontology metrics among a set of concept pairs. The function can be defined over an ontology, or on a single level of an ontology. For an ontology $T(C,R)$, we define its information function as:

$$Info(T) = \sum_{x<y, c_x, c_y \in C} d(c_x, c_y) \tag{6.1}$$

Similarly, we define the information function for an abstraction level L_i as:

$$Info_i(L_i) = \sum_{x<y, c_x, c_y \in L_i} d(c_x, c_y) \tag{6.2}$$

where L_i is the subset of concepts lying at the i^{th} level of an ontology T. For example, in Figure 6.3, node 1 is at level L_1, node 2 and node 5 are at level L_2.

6.2 Assumptions

Given the above definitions, we make the following assumptions about ontologies:

Minimum Evolution Assumption Inspired by the minimum evolution tree selection criterion widely used in phylogeny [HP82], we assume that a good ontology not only minimizes the overall semantic distance among the concepts but also avoids dramatic changes. Thus, construction of a full ontology is proceeded by adding concepts one at a time, which yields a series of partial ontologies. After adding each concept, the best current ontology T^{n+1} from the previous ontology T^n is one that introduces the least changes between the information in the two ontologies:

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

where the *information change function* is

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

Abstractness Assumption In an ontology, concrete concepts usually lay at the bottom of the hierarchy while abstract concepts often occupy the intermediate and top levels. Concrete concepts often represent physical entities, such as "basketball" and "mercury pollution"; while abstract concepts, such as "science" and "economy", do not have a physical form thus we must imagine their existence. This difference suggests that there is a need to treat them differently in ontology learning. Thus we assume that concepts at the same abstraction level have common characteristics and share the same $Info(.)$ function. We also assume that concepts at different abstraction levels have different characteristics; hence they do not necessarily share the same $Info(.)$ function. That is to say,

$$\forall \text{ concept } c \in T, \text{ abstraction level } L_i \in T, c \in L_i \Rightarrow c \text{ uses } Info_i(.).$$

6.3 The Metric-based Ontology Learning Framework

This section presents the metric-based ontology learning framework. Based on the definitions of ontologies, ontology metric and information functions, and minimum evolution assumption and abstractness assumption, this section formulates the task of ontology learning as a multi-criterion optimization and solves the optimization problem by a greedy algorithm. This section also shows how to estimate an ontology metric by learning adaptive weights for underlying feature functions.

6.3.1 The Minimum Evolution Objective

Based on the minimum evolution assumption, we define that the goal of ontology learning is to find the optimal full ontology $\hat{\mathrm{T}}$ such that the information changes are the least since the initial partial ontology T^0, i.e., to find:

$$\hat{T} = \arg\min_{T'} \Delta Info(T^0, T') \tag{6.3}$$

where T' is a full ontology, i.e., the set of concepts equals C.

To find the optimal solution for Equation 6.3, we need to find the optimal concept set $\hat{C}$ and the optimal relation set $\hat{R}$. Since the optimal concept set for a full ontology is always C, the only unknown part left is $\hat{R}$. Thus, Equation 6.3 can be transformed equivalently into:

$$\hat{R} = \arg\min_{R'} \Delta Info(T(C, R'), T^0(S^0, R^0)) \tag{6.4}$$

In this metric-based ontology learning framework, concepts are added incrementally into an ontology. Each insertion of concepts yields a new partial ontology T. By the minimum evolution assumption, the optimal next partial ontology is one gives the least information change. Therefore, the updating function for the set of relations R^{n+1} after a new concept z is inserted can be calculated as:

$$R^{(n+1)} = \arg\min_{R'} \Delta Info(T(S^n \cup \{z\}, R'), T(S^n, R^n)) \tag{6.5}$$

By plugging in the definition of the information change function $\Delta Info(.)$ in Section 6.1 and Equation 6.1, the updating function becomes:

$$R^{(n+1)} = \arg\min_{R'} \| \sum_{c_x, c_y \in S^n \cup \{z\}} d(c_x, c_y) - \sum_{c_x, c_y \in S^n} d(c_x, c_y) \| \tag{6.6}$$

The above updating function can be transformed into a minimization problem:

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

The minimization follows the minimum evolution assumption; hence we call it the *minimum evolution objective*.

6.3.2 The Abstractness Objective

The *abstractness* assumption suggests that concept abstractness should be modeled explicitly by learning separate information functions for concepts at different abstraction levels. We approximate an information function by a linear interpolation of underlying feature functions. Each abstraction level L_i is characterized by its own

information function $Info_i(.)$. The least square fit of $Info_i(.)$ is:

$$\min \sum_i \|Info(L_i) - W_i^T H_i\|^2 \tag{6.7}$$

By plugging Equation 6.2 and minimizing over every abstraction level, we have:

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

where $h_{i,j}(.,.)$ is the j^{th} underlying feature function for concept pairs at level L_i, $w_{i,j}$ is the weight for $h_{i,j}(.,.)$. This minimization follows the abstractness assumption; hence we call it the *abstractness objective*.

Since the 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 combination of heterogenous features.

6.3.3 The Multi-Criterion Optimization Algorithm

In the metric-based ontology learning framework, both *minimum evolution* and *abstractness* objectives need to be satisfied. To optimize multiple criteria, the Pareto optimality needs to be satisfied (Boyd and Vandenberghe, 2004). Since the two optimization problems are both convex, the Pareto optimal can be obtained by scalarization. That is to say, introduce a variable $\lambda \in [0, 1]$ to control the contribution of each objective The multi-criterion optimization function becomes:

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

The above optimization can be solved by a greedy optimization algorithm. At each concept insertion step, it produces a new partial ontology by adding to the existing partial ontology a new concept z, and a new set of relations $R(z, .)$. z is attached to every node in the existing partial ontology; and the algorithm selects the

optimal position indicated by $R(z, .)$, which minimizes the multi-criterion objective function. The algorithm is:

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 greedy algorithm presents a general incremental clustering procedure to construct ontologies. By minimizing the ontology structure changes and modeling concept abstractness at each step, it finds the optimal position of each concept in the ontology.

6.3.4 Estimating Ontology Metric

Learning a good ontology metric is important for the multi-criterion optimization algorithm. In this work, the estimation and prediction of ontology metric are achieved by ridge regression [HTF01].

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

$$d(x, y) = \sum_j w_j h_j(x, y) \tag{6.8}$$

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

The feature functions are generated as mentioned in Section 5.5.

6.4 Evaluation

To evaluate the proposed metric-based ontology learning framework, we use the framework to reconstruct ontologies from WordNet and ODP. The datasets are introduced in Chapter 3.

With the extracted ontologies from WordNet and ODP (Chapter 3), 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 relations between the concepts. The testing data only contain the set of concepts in the corresponding ontology (since we need to find the relations).

6.4.1 Methodology

We evaluate the quality of automatic generated ontologies by comparing them with the gold standards in concepts of precision, recall and F1-measure. F1-measure is calculated as $2 \times P \times R/(P + R)$, where P is *precision*, the percentage of correctly returned relations out of the total returned relations, R is *recall*, the percentage of correctly returned relations out of the total relations in the gold standard.

Table 6.1: System Performance.

WordNet/*is-a*			
System	*Precision*	*Recall*	*F1-measure*
HE	**0.85**	0.32	0.46
GI	n/a	n/a	n/a
PR	0.75	0.73	0.74
ME	0.82	**0.79**	**0.82**
ODP/*is-a*			
System	*Precision*	*Recall*	*F1-measure*
HE	0.31	0.29	0.30
GI	n/a	n/a	n/a
PR	0.60	**0.72**	0.65
ME	**0.64**	0.70	**0.67**
WordNet/*part-of*			
System	*Precision*	*Recall*	*F1-measure*
HE	n/a	n/a	n/a
GI	**0.75**	0.25	0.38
PR	0.68	0.52	0.59
ME	0.69	**0.55**	**0.61**

Because an ontology is in a tree-structure and concepts at the higher levels are harder to obtain due to their higher abstractness, the number of correctly returned hypernym-hyponym pairs, the number of hypernym-hyponym pairs in the gold standard and the total number of returned pairs need to be weighted by concept abstractness. That is to say, we need to 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:

$$\sum_{(i,j)} \mathbf{I}(\text{pair } (i,j) \text{ is returned correctly}) * \frac{max\_level - level(j) + 1}{\sum_{l=1}^{max\_level} l} \tag{6.9}$$

where $\mathbf{I}(.)$ 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 average the system performance across different training and test datasets. For each of the 50 datasets from WordNet hypernyms, WordNet meronyms or ODP hypernyms, we pick 49 of them to generate training data, and test on the remaining dataset. We repeat the process for 50 times, with different training and test sets each time, and report the averaged precision, recall and F1-measure across all 50 runs.

We also group the fifteen features in Section 5.5 into six sets: contextual, co-concurrence, patterns, syntactic dependency, word length difference and definition. Each set is turned on one by one for experiments in the following sections.

6.4.2 Performance of Ontology Learning

In this experiment, we compare the following automatic ontology learning systems: *HE*, the system proposed by Hearst (1992) with 6 hypernym patterns [Hea92]; *GI*, the system proposed by Girju et al. (2003) with 3 meronym patterns [GBM03]; *PR*, the probabilistic framework proposed by Snow et al. (2006) [SJN06]; and *ME*, the metric-based framework proposed in this chapter. To have a fair comparison, for *PR*, we estimate the conditional probability of a relation given the evidence $P(R_{ij}|E_{ij})$, as in [SJN06], by using the same set of features as in *ME*.

Table 6.1 shows precision, recall, and F1-measure of the above systems for WordNet hypernyms (*is-a*), WordNet meronyms (*part-of*) and ODP hypernyms (*is-a*). Bold font indicates the best performance in a column. Note that *HE* is not applicable to *part-of*, and *GI* is not applicable to *is-a*.

Table 6.1 shows that systems using heterogeneous features (*PR* and *ME*) achieve higher F1-measure than systems only using patterns (*HE* and *GI*) with a significant absolute gain of >30%. Generally speaking, pattern-based systems show higher precision and lower recall, while systems using heterogeneous features show lower precision and higher recall. However, when considering both precision and recall, using heterogeneous features is more effective than just using patterns. The proposed system *ME* consistently produces the best F1-measure for all three tasks.

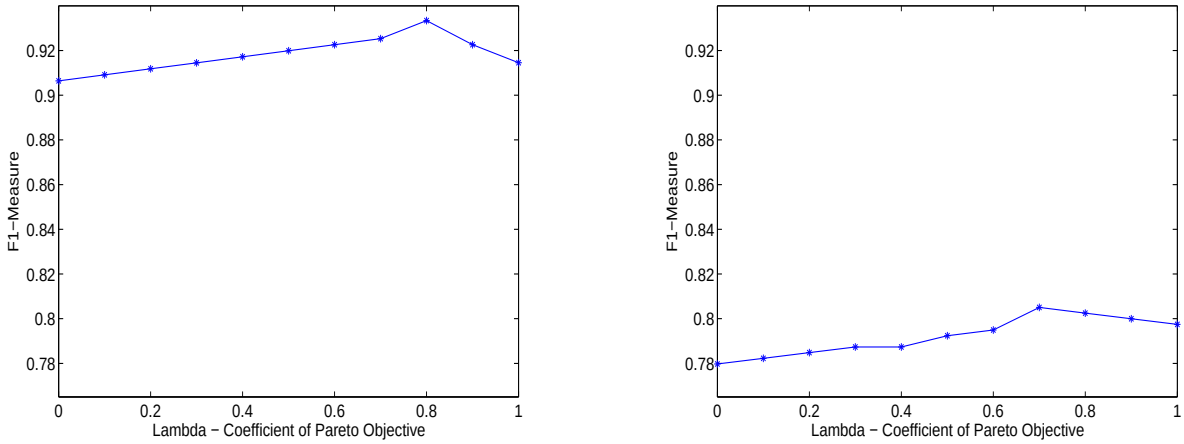
The performance of the systems for ODP/*is-a* is worse than that for WordNet/*is-a*. This may be because there is more noise in ODP than in WordNet. For example, under *artificial intelligence*, ODP has *neural networks*, *natural language* and *academic departments*. Clearly, *academic departments* is not a hyponym of *artificial intelligence*. The noise in ODP interferes with the learning process, thus hurts the performance.

6.4.3 Impact of Concept Abstractness

This experiment studies the impact of modelling concept abstractness on system performance. Figure 6.4(a) and (b) show the changes of system performance when varying the coefficient λ in the Pareto objective as defined in Section 6.3.3. λ is a coefficient to adjust the contributions of minimum evolution objective and abstractness objective. As $\lambda \rightarrow 0$, the system relies more on the abstractness objective whereas as $\lambda \rightarrow 1$, 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 6.4.2, 6.4.4, and 6.4.5.

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.



(a)WordNet (b)ODP

Figure 6.4: Impact of Concept Abstractness.

Table 6.2: F1-measure for Features vs. Relations: WordNet.

Feature	*is-a*	*sibling*	*part-of*	***Benefited Relations***
Contextual	0.21	**0.42**	0.12	*Sibling*
Co-occurrence	**0.48**	**0.41**	**0.28**	*All*
Patterns	**0.46**	**0.41**	**0.30**	*All*
Syntactic	0.22	**0.36**	0.12	*Sibling*
Word Length	0.16	0.16	0.15	*All but limited*
Definition	0.12	0.18	0.10	*Sibling but limited*
All	0.82	0.79	0.61	*All*
Best Features	Co-occurrence, patterns	Contextual, co-occurrence, patterns, syntactic	Co-occurrence, patterns	

6.4.4 Features vs. Relations

This experiment studies the impact of different sets of features on different types of relations. Table 6.2 shows F1-measure of using each set of features alone on ontology learning for WordNet is-a, sibling, and part-of relations. Bold font means a feature set gives a major contribution to the task of automatic ontology learning for a particular type of relation.

Table 6.2 shows that different relations favor different sets of features. Both co-occurrence and lexico-syntactic patterns work well for all three types of relations. It is interesting to see that simple co-occurrence statistics work as good as lexico-syntactic patterns. Contextual features work well for *sibling* relations, but not for *is-a* and *part-of*. Syntactic features also work well for *sibling*, but not for *is-a* and *part-of*. The similar behavior of contextual and syntactic features may be because that four out of five syntactic features (*Modifier*, *Subject*, *Object*, and *Verb* overlaps) are just surrounding context for a concept.

The second last row of 6.2 shows the F1-measures for WordNet *is-a*, *sibling*, and *part-of* relations by using

Table 6.3: F1-measure for Features vs. Abstractness: WordNet/*is-a*.

Feature	L_2	L_3	L_4	L_5	L_6
Contextual	0.29	0.31	**0.35**	**0.36**	**0.36**
Co-occurrence	**0.47**	**0.56**	**0.45**	**0.41**	**0.41**
Patterns	**0.47**	**0.44**	**0.42**	**0.39**	**0.40**
Syntactic	0.31	0.28	**0.36**	**0.38**	**0.40**
Word Length	0.16	0.16	0.16	0.16	0.16
Definition	0.12	0.12	0.12	0.12	0.12

Table 6.4: F1-measure for Features vs. Abstractness: ODP/*is-a*.

Feature	L_2	L_3	L_4	L_5	L_6
Contextual	**0.30**	**0.30**	**0.33**	**0.29**	**0.29**
Co-occurrence	**0.34**	**0.36**	**0.34**	**0.31**	**0.31**
Patterns	0.23	0.25	**0.30**	**0.28**	**0.28**
Syntactic	0.18	0.18	0.23	**0.27**	**0.27**
Word Length	0.15	0.15	0.15	0.14	0.14
Definition	0.13	0.13	0.13	0.12	0.12

all features. We notice a significant gain in F1-measure by using all features than using individual features. It indicates that combination of heterogeneous features gives more rise to the system performance than a single set of features does.

6.4.5 Features vs. Abstractness

This section studies the impact of different sets of features on concepts at different abstraction levels. In the experiments, F1-measure is evaluated for concepts at each level of an ontology, not the whole ontology. Table 6.3 and Table 6.4 demonstrate F1-measure of using each set of features alone on each abstraction levels. Columns 2-6 are indices of the levels in an ontology. The larger the indices are, the lower the levels. Higher levels contain abstract concepts, while lower levels contain concrete concepts. L_1 is ignored here since it only contains a single concept, the root. Bold font indicates good performance in a column.

Both Table 6.3 and Table 6.4 show that abstract concepts and concrete concepts favor different sets of features. In particular, contextual, co-occurrence, pattern, and syntactic features work well for concepts at $L_4 - L_6$, i.e., concrete concepts; co-occurrence works well for concepts at $L_2 - L_3$, i.e., abstract concepts. This difference indicates that concepts at different abstraction levels have different characteristics; it confirms our *abstractness assumption* in Section 6.1.

We also observe that for abstract concepts in WordNet, patterns work better than contextual features; while for abstract concepts in ODP, the conclusion is the opposite. This may be because that WordNet has a richer vocabulary and a more rigid definition of hypernyms, and hence is-a relations in WordNet are recognized more effectively by using lexico-syntactic patterns; while ODP contains more noise, and hence it favors features requiring less rigidity, such as the contextual features generated from the Web.

6.5 Summary

This chapter presents a novel metric-based ontology learning framework which incrementally clusters concepts and transforms the task of ontology learning into a multi-criteria optimization based on minimization of ontology structures and modeling of concept abstractness. The experiments show that our framework is effective; it achieves higher F1-measure than three state-of-the-art systems.

This chapter also studies which features are the best for different types of relations. The experiments show that co-occurrence and patterns are good features for common relations, such as is-a, sibling, and part-of. Contextual and syntactic features are only good for sibling relations. Moreover, this chapter studies which features are the best for concepts at different abstraction levels. The experiments show that abstract concepts and concrete concepts favor different sets of features. Contextual, co-occurrence, patterns, and syntactic features work well for concrete concepts. Co-occurrence works well for abstract concepts; the performance of patterns and contextual features for abstract concepts depends on data.

Most prior work uses a single rule or feature function for automatic ontology learning at all levels of abstraction. Our work is a more general framework which allows a wider range of features and different metric functions at different abstraction levels. This more general framework has the potential to learn more complex ontologies than previous approaches.

Chapter 7

Dissertation Research

7.1 A Unified Personal Ontology Learning Framework

The previous chapters presented the techniques for concept extraction (Chapter 4) as well as two frameworks for relation formation for the task of personal ontology learning. The first relation formation framework is the human-guided ontology learning framework (Chapter 5), which integrates both human efforts and machine learning techniques to facilitate personal ontology learning. The second framework is the metric-based ontology learning framework (Chapter 6), which addresses several existing problems, including limited use of features, no controlling of concept abstractness, and unknown cluster labels, in ontology learning.

Comparing the two frameworks, we find that human-guided ontology learning framework does not handle well the problems of unknown cluster labels and no controlling of concept abstractness. On the other hand, the metric-based ontology learning framework does not take advantage of human guidance to allow personalization in ontology learning. In the dissertation research, we need to integrate the two frameworks presented in the proposal into one unified learning framework. In particular, we need to take care both human guidance, i.e., the personal preferences, as well as modelling of concept abstractness, in the new unified learning framework.

Moreover, in the unified learning framework, it is necessary to handle long distance relations properly. The work presented in this proposal is mainly based on immediate relations of concepts in an ontology. For example, we only study parent/child relation in a hypernym ontology; other relations, such as grandparent/grandchild relation, ancestor/descender relation, and cousin relation, are not studied in a hypernym ontology. These *long distance relations* are not examined in the thesis proposal. In the dissertation research, we need to extend the work to deal with such long distance relations to ensure greater global consistency in an ontology. We need to design a more general framework to allow handling of multiple relations, including both immediate relations and long distance relations, simultaneously.

Figure 7.1 shows the design of the proposed unified personal ontology learning framework. The framework can be divided into three parts.

The first part is the inputs to the unified personal ontology learning framework. These inputs include the concepts extracted from a corpus, training data with labels, human guidance, and expert-crafted rules. The

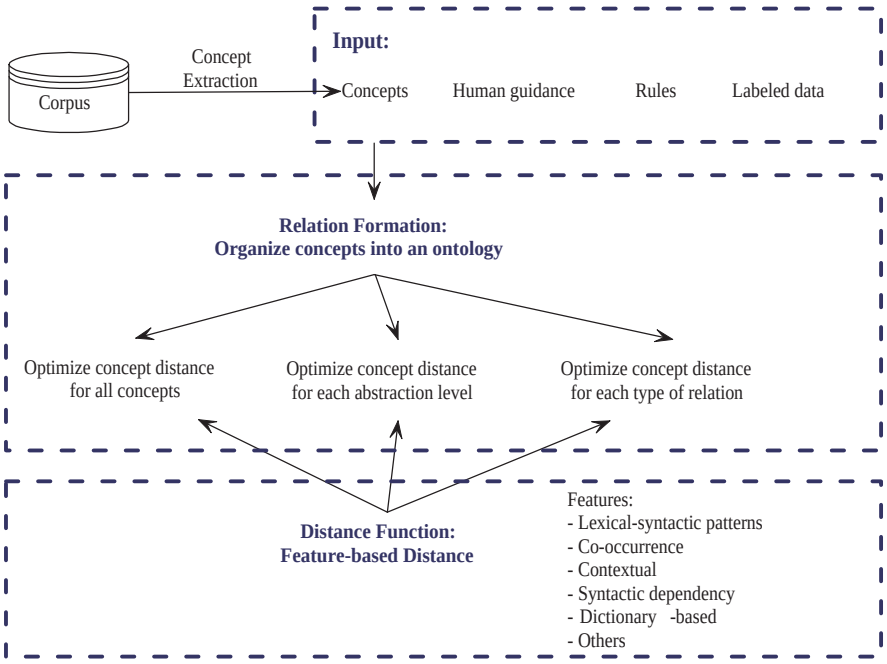


Figure 7.1: A Unified Personal Ontology Learning Framework.

multiple forms of inputs make the proposed framework flexible enough to integrate useful information from a wide range of resources.

The second part is a relation formation process via a multi-criterion optimization algorithm to discover the correct relations among concepts and to organize these concepts into ontologies. The optimization algorithm is similar to what is presented in the metric-based ontology learning framework. This optimization algorithm optimizes the ontology structure through optimizing concept distances from three aspects, including immediate pairwise distances between all concepts, distances for concepts at each abstraction level, and distances for concepts of different types of semantic relations. In Chapter 6, we have demonstrated the optimization from the first two aspects. In the proposed unified framework, we will extend the existing work by adding modelling of concept distances for each type of relations, including "parent/child", "grandparent/grandchild", "ancestor/descender", "cousin" relations, etc.

The third part is a distance function which quantitatively measures semantic relations between two concepts. In both human-guided ontology learning and metric-based ontology learning frameworks, we used a feature-based distance function which incorporates a variety of underlying semantic features (Section 5.5). Those features vary from lexical-syntactic patterns, co-occurrence, contextual, knowledge-based features to complicated features like parse-tree-based features. The features will again be used in the proposed unified framework. However, we will also explore some other interesting semantic features mentioned in the literature, or invent some new interesting features.

The design of this unified personal ontology learning framework allows us to address *personalization*, *concept abstractness*, *multiple types of relations*, and *a even richer group of semantic features*, for the task of personal

ontology learning.

7.2 Human Impact vs. Task Impact

Personal ontology learning involves both human efforts and machine learning techniques. It is personalized and task-specific. For a particular task, such as organizing the literature of natural language processing (NLP), if the task is performed by different individuals, personal ontology learning probably yields different ontologies. However, we would like to ask - "Are there any common characteristics across different individuals for constructing this NLP literature ontology"? Likewise, for a particular person, if he/she is given different tasks, personal ontology learning probably also yields different ontologies. We also would like to know - "Are there any common features which capture the habit or characteristics of how this person organizes concepts for different tasks"? The new research proposed for this dissertation provides an opportunity to study the human impact and task impact on personal ontology learning, as well as the interactions between humans and tasks.

For example, suppose there are three individuals A, B and C. And there are also three tasks, namely finding a good kindergarten, hiring a good wedding videographer, and buying a used car. The proposed work will study the ontologies constructed for each tasks by each individual. The user study will provide true/false answers for the following hypotheses:

- Hypothesis One:
  There are common patterns for a particular person to organize ontologies for different tasks.
- Hypothesis Two:
  The common patterns for a particular person to organize ontologies for different tasks can be transferred from one task to another.
- Hypothesis Three:
  There are common patterns for a particular task to be organized into an ontology by different individuals.
- Hypothesis Four:
  The common patterns for a task to be organized into an ontology by different individuals can be transferred from one individual to another.

Hypothesis one and Hypothesis three address the human impact and task impact on personal ontology learning. The answers to these two hypotheses may uncover interesting human behaviors and patterns to organize materials, as well as interesting task-related patterns to construct ontologies.

Hypothesis two and Hypothesis four address knowledge transfer among humans and knowledge transfer among tasks during personal ontology learning. If the answers to these two hypotheses are positive, knowledge transfer techniques can be used to further accelerate personal ontology learning since the machine can learn not only from a particular individual/task but from multiple individuals/tasks. For example, a simple approach of knowledge transfer (based on Hypothesis two) could be as follows: Based on two training tasks, for instance, the kindergarten and videographer tasks, features which are highly ranked in both tasks by the same individual can be considered as common features for personal ontology learning across all tasks. In another word, these common features are

considered as task-independent features. Other features may be task-dependent features and may not be suitable to apply to other tasks.

7.3 User Studies

As a human computer interaction system, the proposed unified personal ontology learning framework also needs to be tested through user studies. Moreover, the hypotheses in Section 7.2 need to be tested through extensive user studies.

We plan to hire 10-25 users in our study. The users will be instructed to construct personal ontologies. For each user, he/she needs to finish 3-4 ontology construction tasks. During each session, users will be divided into two groups. One group is the manual group, who will construct ontologies totally based on manual efforts. Another group is the interactive group, who will construct ontologies by using our interactive system. Each user will be assigned to different groups for different tasks so that every user will have experience on both manual ontology construction and interactive ontology construction. We will ask each user to compare and evaluate the two kinds of constructions in the end. We will also ask the users come back one or two weeks later to perform the same task and measure their self-agreement.

Based on our previous experience, user studies are time-consuming and intensive. We will either collaborate with a coding lab or hire graduate students to conduct the studies.

As an alternative, we are investigating the possibility of hiring online participants through Amazon Mechanical Turk[1].

7.4 Automatic Cluster Labeling

There is considerable prior research on hierarchical clustering algorithms and their applications in information retrieval and data mining research. However, less attention has been paid to creating good cluster labels. Cluster names created automatically often either fail to provide a comprehensive label of the cluster, or consist of lists of concepts from which a person must infer a more general label. Our goal is to select concise cluster labels that are similar to what a person might create manually.

In the human-guided ontology learning framework (Chapter 5), when K-medoids algorithm creates new clusters for the ungrouped concepts, these new clusters are nameless. These newly-created clusters need meaningful labels. The label for a new cluster should be general enough to cover the scope of all of its child concepts, and be specific enough to just cover that of them. We used the Google search engine as the provider of general knowledge (Details see Section 5.6).

There are limitations in our exisiting algorithm. First of all, it relies too much on the search results provided by Google. Since formed by the cluster members, the queries are long; Google may not return anything, which fails the existing algorithm. Moreover, the existing algorithm is sensitive to the stopword list. If a good cluster label is part of the stopword list, this good label has no chance to be selected from the retrieved web documents or snippets. On the other hand, if the stopword list is not extensive enough to exclude a general word, then this

[1] https://www.mturk.com/mturk/.

general word is likely to be the most frequent term in the returned top 10 snippets, and hence is selected as the cluster label. Both situations are undesirable. Furthermore, the algorithm is sensitive to the threshold of number of snippets. If this threshold is set as 50, the most frequent term will probably be different from the most frequent term when this threshold is set as 10.

Due to the above limitations, a more reliable algorithm is needed. We propose to employ a combination of multiple web search engines as well as a local search engine (Indri [SMTC05]) which creates index for the related text documents of an ontology. The search results from both the local search engine and the Web search engines will be merged and ranked.

We will also employ glosses from existing dictionaries like WordNet, Wiktionary, and Oxford to generate good cluster labels.

The stopword list and top document threshold will be automatically learned by machine learning algorithms. The possible machine learning algorithms can be used include ridge regression, rank SVM, and decision trees.

Chapter 8

Expected Contributions of This Dissertation

The research discussed in this proposal and the scheduled work in the dissertation research is the first effort to study *personal ontology learning*. This research is also a significant piece of work for *ontology learning* in general. The proposed human-guided ontology learning framework and metric-based ontology framework address various aspects in the task of ontology learning. These aspects include *concept abstractness*, *use of heterogenous semantic features*, and *integration of human guidance*. The scheduled research on the unified personal ontology learning framework unifies and presents solutions to all these aspects. The scheduled research integrates techniques in machine learning, natural language processing, information retrieval into one framework to advance the research of ontology learning.

The research in the scheduled work is also the first effort to study human-task interaction as well as knowledge transfer during ontology learning among different persons and that among different tasks. This will be an important step in the fields of both Human Computer Interaction and Transfer Learning. Our extensive user studies will provide first-hand materials for research in personalized tasks for a broad range of applications.

In summary, the research in this dissertation is the first step of *personal ontology learning*, and an important step forward of *ontology learning*. This work addresses important problems of ontology learning, especially considers how to better model these problems with better theoretical foundations. This work not only integrates techniques from various research fields but also contributes to the development of those related research fields. The better theoretical foundation, the more extensive empirical experiments and user studies, and the better modelling of various aspects of ontology learning in this new research show a bright future of *personal ontology learning*.

Chapter 9

Schedule

- Oct 2009 - Dec 2009
  Further study: propose and study new algorithms for the unified framework; design and implement new algorithms to handle long distance relations; small scale evaluation.
- Jan 2010 - Apr 2010
  Further study: propose and study new methodology and evaluation metrics to study human impact and task impact on personal ontology learning and their interactions; user study and evaluation for unified framework and human-task interactions.
- May 2010 - Jun 2010
  Further Study: propose and study new algorithms for the automatic cluster labelling; a combination of multiple online/local search engine results; use glosses from existing dictionaries; learn an adaptive threshold for the number of snippets.
- Jul 2010 - Nov 2010
  Analyze the experimental and user study results of the new research; summarize and write up the thesis.

Appendix A

Example Ontologies

In order to better demonstrate the ontology construction task presented in this work, we show three actual ontologies developed by human guided ontology learning framework. The first two ontologies (shown in Figure A.1 and A.2) were created by two different users manually. The third ontology (shown in Figure A.3) was created by another user who worked interactively with OntoCop. The dataset which these three ontologies were developed for is "Tri".

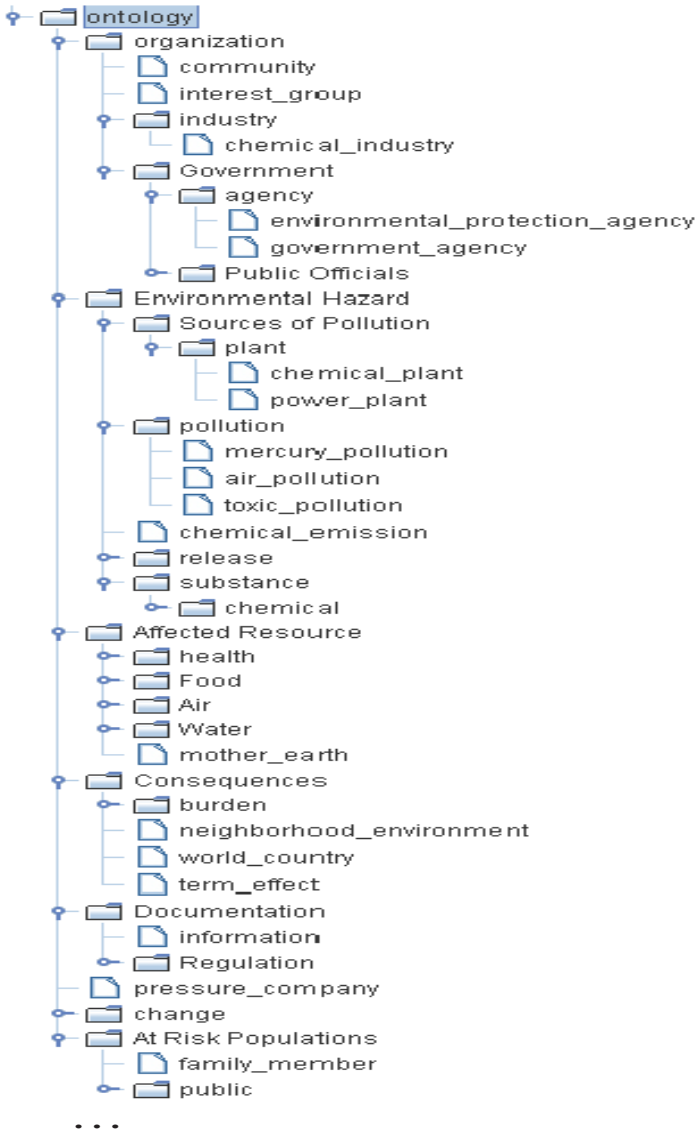


Figure A.1: Ontology Created by User 1 (Manual).

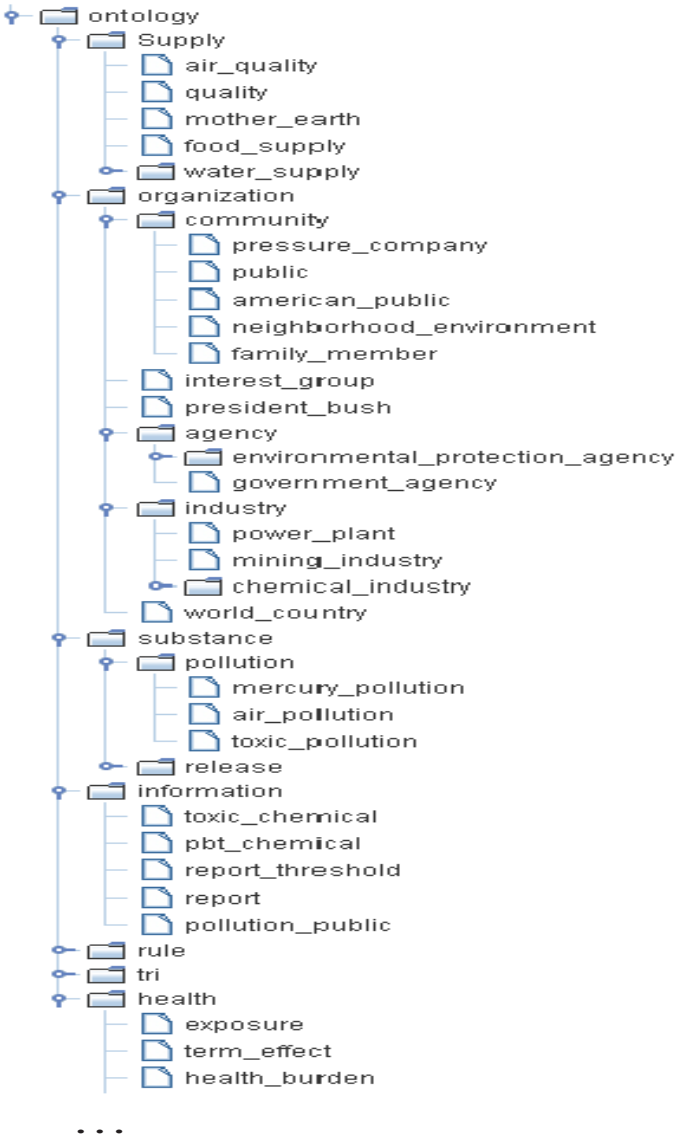


Figure A.2: Ontology Created by User 3 (Manual).

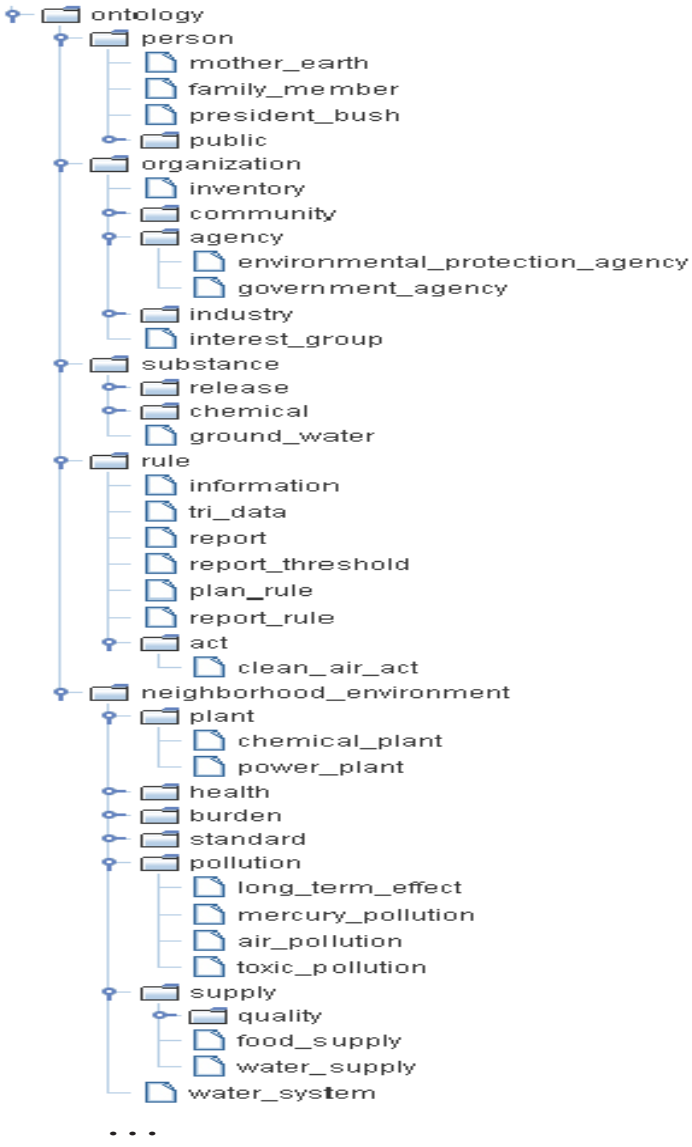


Figure A.3: Ontology Created by User 5 (Interactive).

Bibliography

[Ash00] M. Ashburner. Gene ontology: Tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.

[BC99] M. Berland and E. Charniak. Finding parts in very large corpora. In *Proceedings of the Annual Meeting for the Association for Computational Linguistics (ACL 1999)*, 1999.

[BCM05] P. Buitelaar, P. Cimiano, and B. Magnini. Ontology learning from text: Methods, evaluation and applications. In *Volume 123 Frontiers in Artifi-cial Intelligence and Applications.*, 2005.

[Bha06] Rajendra Bhatia. *Positive Definite Matrices (Princeton Series in Applied Mathematics)*. Princeton University Press, December 2006.

[Blo05] Eva 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.

[BM07] R. Bunescu and R. Mooney. Learning to extract relations from the web using minimal supervision. In *Proceedings of the 45th Annual Meeting for the Association for Computational Linguistics (ACL 2007)*, 2007.

[BPd+92] P. Brown, V. D. Pietra, P. deSouza, J. Lai, and R. Mercer. Class-based ngram models for natural language. In *Computational Linguistics,18(4):468-479*, 1992.

[Car96] J. Carletta. Assessing agreement on classification tasks: The kappa statistic. In *Computational Linguistics, 22(2):249-254*, 1996.

[Car99] Sharon 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 (ACL 1999)*, 1999.

[CGJ96] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

[CHS04] Philipp Cimiano, Andreas Hotho, and Steffen Staab. Comparing conceptual, divisive and agglomerative clustering for learning taxonomies from text. In *Proceedings of the European Conference on Artificial Intelligence*, 2004.

[Coh60] J. Cohen. A coefficient of agreement for nominal scales. In *Educational and Psychological Measurement, 20, 37-46*, 1960.

[CP04] T. Chklovski and P. Pantel. Verbocean: mining the web for fine-grained semantic verb relations. In *Proceedings of the Annual Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, 2004.

[CSV04] F. Colace, M. De 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.

[CV05a] P. Cimiano and J. Volker. Towards large-scale, open-domain and ontology-based named entity classification. In *Proceedings of Recent Advances in Nat-ural Language Processing, pages 166-172*, 2005.

[CV05b] Philipp Cimiano and Johanna 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.

[CV08] Sonia Chernova and Manuela Veloso. Teaching multi-robot coordination using demonstration of communication and state sharing. In *AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, 2008.

[CW07] P. Cimiano and J. Wenderoth. Automatic acquisition of ranked qualia structures from the web. In *Proceedings of the 45th Annual Meeting for the Association for Computational Linguistics (ACL 2007)*, 2007.

[DH02] Melania Degeratu and Vasileios Hatzivassiloglou. Building automatically a business registration ontology. In *Proceedings of the 2nd National Conference on Digital Government Research*, 2002.

[DR06] D. Davidov and A. Rappoport. Efficient unsupervised discovery of word categories using symmetric patterns and high frequency words. In *Proceedings of the 44th Annual Meeting for the Association for Computational Linguistics (ACL 2006)*, 2006.

[ECD+05] O. Etzioni, M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. Unsupervised named-entity extraction from the web: an experimental study. In *Artificial Intelligence, 165(1):91-134, June*, 2005.

[Fel98] Christiane Fellbaum. *WordNet:An Electronic Lexical Database*. MIT Press, 1998.

[FMG05] Blaz Fortuna, Dunja Mladenic, and Marko Grobelnik. Semi-automatic construction of topic ontology. In *Conference on Data Mining and Data Warehouses*. SiKDD, 2005.

[FPL93] N. Tishby F. Pereira and L. Lee. Distributional clustering of english words. In *Proceedings of the 31th Annual Meeting for the Association for Computational Linguistics (ACL 1993)*, 1993.

[GBM03] R. Girju, A. Badulescu, and D. Moldovan. Learning semantic constraints for the automatic discovery of part-whole relations. In *Proceedings of the Human Language Technology Conference (HLT 2003)*, 2003.

[GBM06] R. Girju, A. Badulescu, and D. Moldovan. Automatic discovery of part-whole relations. In *Computational Linguistics, 32(1): 83-135*, 2006.

[Gre84] P. M. Greenfield. Theory of the teacher in learning activities of everyday life. In *Everyday cognition: its development in social context, Harvard University Press*, 1984.

[Har54] Z. Harris. Distributional structure. In *Word, 10(23): 146-162s*, 1954.

[Hea92] M.A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING 1992)*, 1992.

[HM06] Yifen Huang and Tom Mitchell. Text clustering with extended user feedback. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR, 2006.

[HM07] Yifen Huang and Tom Mitchell. A framework for mixed-initiative clustering. In *North East Student Colloquium on Artificial Intelligence (NESCAI 2007)*, 2007.

[HM08] Yifen Huang and Tom Mitchell. Exploring hierarchical user feedback in email clustering. In *Enhanced Messaging Workshop (AAAI 2008)*, 2008.

[HP82] M. D. Hendy and D. Penny. *Branch and bound algorithms to determine minimal evolutionary trees.* Mathematical Biosciences 59: 277-290 1982.

[HTF01] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer-Verlag, 2001.

[HZ09] Shuming Shi Ji-Rong Wen Huibin Zhang, Mingjie Zhu. Employing topic models for pattern-based semantic class discovery. In *Proceedings of the 47th Annual Meeting for the Association for Computational Linguistics (ACL 2009)*, 2009.

[KAB04] Lobna Karoui, Marie-Aude Aufaure, and Nacra Bennacer. Ontology discovery from web pages: Application to tourism. In *In the Workshop of Knowledge Discovery and Ontologies*, 2004.

[KKSM05] Andruid Kerne, Eunyee Koh, Vikram Sundaram, and J. Michael Mistrot. Generative semantic clustering in spatial hypertext. In *DocEng '05: Proceedings of the 2005 ACM symposium on Document engineering*, 2005.

[KRH08] Zornitsa Kozareva, Ellen Riloff, and Eduard Hovy. Semantic class learning from the web with hyponym pattern linkage graphs. In *Proceedings of the 46th Annual Meeting for the Association for Computational Linguistics (ACL 2008)*, 2008.

[KW02] Latifur Khan and Lei Wang. Automatic ontology derivation using clustering for image classification. In *In Proc. of 8th International Workshop on Multimedia Information Systems*, 2002.

[LC03] D. J. Lawrie and W. B. Croft. Generating hierarchical summaries for web searches. In *Proceedings of the Annual International ACM Special Interest Group On Information Retrieval (SIGIR) Conference*, 2003.

[Len95] D. Lenat. Cyc: a large-scale investment in knowledge infrastructure. *Communications of the Association for Computing Machinery*, 38(11), 1995.

[Lin98] D. Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 1998)*, 1998.

[LZQZ03] D. Lin, S. Zhao, L. Qin, and M. Zhou. Identifying synonyms among distributionally similar words. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI 2003)*, 2003.

[Mac67] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *volume 1 of Proceedings of the Fifth Berkeley Symposium on Mathematical statistics and probability*, 1967.

[Mah36] P. C. Mahalanobis. On the generalised distance in statistics. In *Proceedings of the National Institute of Sciences of India 2 (1): 4955*, 1936.

[Man02] G. S. Mann. Fine-grained proper noun ontologies for question answering. In *Proceedings of SemaNet'02:Building and Using Semantic Networks*, 2002.

[MST+05] Richard Maclin, Jude W. Shavlik, Lisa Torrey, Trevor Walker, and Edward W. Wild. Giving advice about preferred actions to reinforcement learners via knowledge-based kernel regression. In *AAAI*, 2005.

[NJH01] S. J. Nelson, D. Johnston, and B. L. Humphreys. *Relationships in Medical Subject Headings*. Kluwer Academic Publishers, 2001.

[NM03] Monica N. Nicolescu and Maja J. Mataric. Natural methods for robot task learning: instructive demonstrations, generalization and practice. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, 2003.

[PL02] P. Pantel and D Lin. Discovering word senses from text. In *Proceedings of 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD 2002)*, 2002.

[PP06] P. Pantel and M. Pennacchiotti. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the 44th Annual Meeting for the Association for Computational Linguistics (ACL 2006)*, 2006.

[PR04] P. Pantel and D. Ravichandran. Automatically labeling semantic classes. In *Proceedings of Human Language Technology conference / North American chapter of the Association for Computational Linguistics annual meeting (HLT/NAACL 2004)*, 2004.

[PRH04] P. Pantel, D. Ravichandran, and E. Hovy. Towards terascale knowledge acquisition. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING 2004)*, 2004.

[RC98] B. Roark and E. Charniak. Noun-phrase co-occurrence statistics for semi-automatic semantic lexicon construction. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and COLING (ACL/COLING 98)*, 1998.

[RF07] B. Rosenfeld and R. Feldman. Clustering for unsu-pervised relation identification. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management (CIKM 2007)*, 2007.

[RH02] D. Ravichandran and E. Hovy. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting for the Association for Computational Linguistics (ACL 2002)*, 2002.

[RS97] E. Riloff and J. Shepherd. A corpus-based approach for building semantic lexicons. In *Proceedings of the Annual Conference on Empirical Methods in Natural Language Processing (EMNLP 1997)*, 1997.

[Sab04] Marta Sabou. Extracting ontologies from software documentation. In *Workshop on Ontology Learning and Population, European Conference on Artificial Intelligence*, 2004.

[SC99] Mark Sanderson and Bruce Croft. Deriving concept hierarchies from text. In *Proceedings of the 22nd Annual International ACM Special Interest Group On Information Retrieval (SIGIR) Conference*, 1999.

[SC00] Greg Schohn and David Cohn. Less is more: Active learning with support vector machines. In *Proc. 17th International Conf. on Machine Learning*, pages 839–846. Morgan Kaufmann, San Francisco, CA, 2000.

[SHTC04] I. Szpektor, I. Dagan H. Tanev, and B. Coppola. Scaling web-based acquisition of entailment relations. In *Proceedings of the Annual Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, 2004.

[SJN05] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In *The 19th Annual Conference on Neural Information Processing Systems*, 2005.

[SJN06] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Semantic taxonomy induction from heterogenous evidence. In *The Joint Conference of International Committee on Computational Linguistics (COLING) and the Association for Computational Linguistics (ACL)*, 2006.

[SK01] L. Steels and F. Kaplan. Aibo's first words: The social learning of language and meaning. *Evolution of Communication*, 4(1):3–32, 2001.

[SMTC05] Trevor Strohman, Donald Metzler, Howard Turtle, and W. Bruce Croft. Indri: A language model-based search engine for complex queries, 2005. poster presentation.

[SWGM05] Marta Sabou, Chris Wroe, Carole Goble, and Gilad Mishne. Learning domain ontologies for web service descriptions: An experiment in bioinformatics. In *Proceedings of the 14th International Conference on World Wide Web*, 2005.

[TB06] Andrea Lockerd Thomaz and Cynthia Breazeal. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *AAAI*, 2006.

[TC09] Andrea L. Thomaz and Maya Cakmak. Learning about objects with human teachers. In *HRI '09: Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, 2009.

[TLBS03] P. Turney, M. Littman, J. Bigham, and V. Shnayder. Combining independent modules to solve multiple-choice synonym and analogy problems. In *Proceedings of the International Conference of Recent Advances in Natural Language Processing (RANLP 2003)*, 2003.

[Tut01] W. T. Tutte. Graph theory. In *Cambridge University Press*, 2001.

[TWH00] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a dataset via the gap statistic. In *Tech. Rep. 208, Dept. of Statistics, Stanford University*, 2000.

[WD02] D. Widdows and B. Dorow. A graph model for unsupervised lexical acquisition. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING 2002)*, 2002.

[WVH06] Yimin Wang, Johanna Volker, and Peter Haase. Towards semi-automatic ontology building supported by large-scale knowledge acquisition. In *In AAAI Fall Symposium On Semantic Web for Collaborative Knowledge Acquisition*, 2006.

[YC06] Hui Yang and Jamie Callan. Near-duplicate detection by instance-level constrained clustering. In *Proceedings of the 29th Annual International ACM Special Interest Group On Information Retrieval (SIGIR) Conference*, 2006.

[YC08a] Hui Yang and Jamie Callan. Learning the distance metric in a personal ontology. In *Workshop on Ontologies and Information Systems for the Semantic Web of 17th Conference on Information and Knowledge Management (CIKM2008)*, 2008.

[YC08b] Hui Yang and Jamie Callan. Ontology generation for large email collections. In *Proceedings of the Eighth National Conference on Digital Government Research*, 2008.

[YC09a] Hui Yang and Jamie Callan. Feature selection for automatic taxonomy induction (poster description). In *Proceedings of the 32nd Annual International ACM Special Interest Group On Information Retrieval (SIGIR) Conference*, 2009.

[YC09b] Hui Yang and Jamie Callan. A metric-based framework for automatic taxonomy induction. In *Proceedings of the 47th Annual Meeting for the Association for Computational Linguistics (ACL 2009)*, 2009.