Personalized Concept Hierarchy Construction

Hui Yang

CMU-LTI-11-018

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:

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

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
In Language and Information Technologies*

Abstract

A concept hierarchy is a set of concepts and relations between those concepts. Since ancient times, concept hierarchies have been used to organize and access information. In some situations, task-specific and user-specific concept hierarchies are necessary to allow an overview and easy access a large set of documents. For example, in regulatory reforms, rule-makers in government regulatory agencies must quickly identify and respond to issues raised in public comments. A concept hierarchy constructed for a set of public comments hierarchically organizes the comments and a user is able to easily "drill down" into documents that discuss a specific topic.

Particularly, this dissertation addresses how to construct concept hierarchies from text collections automatically or with a-human-in-the-loop. The novel metric-based concept hierarchy construction framework transforms concept hierarchy construction into a multi-criterion optimization problem. It incrementally clusters concepts based on minimum evolution of hierarchy structure, as well as optimization derived from the modeling of concept abstractness and concept coherence. Moreover, this dissertation represents the semantic distance between concepts as a wide range of features, each of which corresponds to a state-of-the-art concept hierarchy construction technique, such as lexico-syntactic pattern, contextual information, and co-occurrence. The use of multiple features allows a further study of the interaction between features and different types of semantic relations as well as the interaction between features and concepts at different abstraction levels.

Besides the automatic framework for concept hierarchy construction, this dissertation also proposes an effective human-guided concept hierarchy construction framework to address personalization by learning from periodic manual guidance and directing the learned models towards personal preferences. Through human-computer interactions, the human and the machine work together to organize concepts into hierarchies. The machine's predictions not only save the user's effort but also make sensible suggestions to assist the user. This is one of the first works of real-time machine learning for organizing personalized and task-specific information in an interactive paradigm.

This dissertation also studies user behaviors during concept hierarchy construction. It explores whether people create concept hierarchies more quickly or more consistently using the proposed frameworks, whether there are consistent dataset-specific or user-specific differences in the hierarchies that people construct, whether people are self-consistent, and how these factors interact with different construction methods. The user study elaborates that dataset difficulty is a major factor affecting how people organize information into concept hierarchies. It also reveals that people are quite self-consistent in building hierarchies. This novel finding provides foundations to study the differences in concept hierarchy construction behaviors between individuals.

Last but not least, the dissertation proposes a novel similarity metric for measuring hierarchy similarity. Fragment-based Similarity (FBS) employs a unique bag-of-word representation for hierarchies and takes a fragment-based view to calculate hierarchy similarity. FBS well approximates tree edit distance and greatly improves tree edit distance's efficiency from NP-hard to only $O(n^3)$ and $O(n)$ if pairwise node similarities are pre-calculated.

The research in this dissertation is an important step forward of concept hierarchy construction. It addresses important problems of concept hierarchy construction, 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 constructing personal concept hierarchies.

Acknowledgement

It is my great pleasure to express my deep and sincere gratitude to those who made this PhD dissertation possible.

Foremost, I am deeply grateful to my advisor, Professor Jamie Callan, for his continuous support and wonderful guidance throughout my entire PhD study. Professor Jamie Callan is a great advisor who is inspiring, perceptive, and patient. Professor Jamie Callan gave me the greatest encouragement to explore the fabulous research area of Information Retrieval and tremendously help me to focus on the essential things. Most importantly, I learn from Professor Jamie Callan how to be a rigorous scholar. To me, Professor Jamie Callan is not only an academic advisor, but also a role model and a lifetime mentor.

Besides my advisor, I wish to express my sincerest gratitude to the rest of my thesis committee: Professor Jaime Carbonell, Professor Christos Faloutsos, and Professor Eduard Hovy, for their valuable advice and insightful comments. I greatly benefit from their encouragement, brilliant ideas and high-standard questions.

I would also express my warmest gratitude to Professor Tat-Seng Chua who introduced me to the wonderful field of Information Retrieval and gave me important guidance and encouragement during my initial attempts in academic research.

I am also indebted to many collaborators and friends at Carnegie Mellon University, National University of Singapore, University of Pittsburgh, Microsoft and elsewhere for their great support and kind help. I benefit enormously from those extensive discussions, lunch time chats, and practice talks. I extend my thanks to Professor Yiming Yang, Professor Stuart Shulman, Anton Mityagin, Krysta Svore, Professor Jingtao Wang, Professor Milos Hauskrecht, Professor Scott Falman, Professor Noah Smith, Professor Aarti Singh, Dr. Alex Hauptmann, Professor Lori Levin, Professor Hwee-Tou Ng, Professor Chin-Hui Lee, Dr. Jon

Elsas, Professor Jaime Arguello, Dr. Yifen Huang, Dr. Vasco Calais Pedro, Professor Jiong Sun, Dr. Kaimin Chang, Yi-Chia Wang, Dr. Jonathan Chung-Kuan Huang, Dr. Meryem Pinar Donmez, Andreas Zollmann, Professor Luo Si, Professor Yi Zhang, Dr. Fan Li, Dr. Jian Zhang, Yi Chang, Chuang Wu, Dr. Jie Lu, Dr. Paul Ogilvie, Dr. Kevyn Collins-Thompson, Dr. Yanjun Qi, Professor Yan Liu, Dr. Rong Yan, Yangbo Zhu, Pucktada Treeratpituk, Le Zhao, Ni Lao, Anagha Kulkarni, Dr. Shinjae Yoo, Dr. Abhimanyu Lad, Dr. Lingyun Gu, Dr. Wen Wu, Justin Betteridge, Frank Lin, Andrew Schlaikjer, Hideki Shima, Dr. Tien-Ho Lin, Dr. Oznur Tastan, David Pane, Mark Hoy, Thi Truong Avrahami, Dr. Michal Valko, Dr. Richard Pelikan, and many many more.

I owe my warmest thanks to my entire family for their love and understanding. My parents always provide me helpful and timely advice and help me get through the difficult times. My husband is the one who is always be my side and always believes in me. I am so grateful to his unconditional love and enormous support. Without them, this dissertation would not be possible. I also thank my daughter Victoria, who just turned to two-year old, for not crying too much when mom had to work.

I dedicate this dissertation to my family.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In our daily work and life, the following situation should be familiar to many of us: given a large set of documents, which could be a long list of search results, a pile of research papers, or a set of lawsuit documentation, a user needs to find out not just one piece of precise, highly accurate answer to fulfil a *lookup* information need, instead, she needs to *learn* and *investigate* a comprehensive set of information for a complex task-oriented information need.

This kind of information needs correspond to a broad spectrum of representations for the set of complete and comprehensive information. At one end of this spectrum is the simplest representation that could be a list of all relevant documents about a concept or a search query. At another end is the most sophisticated representation that could be a comprehensive summary and analysis, aiming to provide the user an analytic result with inferencing and reasoning and to supply decisions as an intelligent decision engine. Somewhere between the two ends is presenting the related concepts and issues mentioned in the document set in an organized form, in particular, a concept hierarchy, to ease information triage and access.

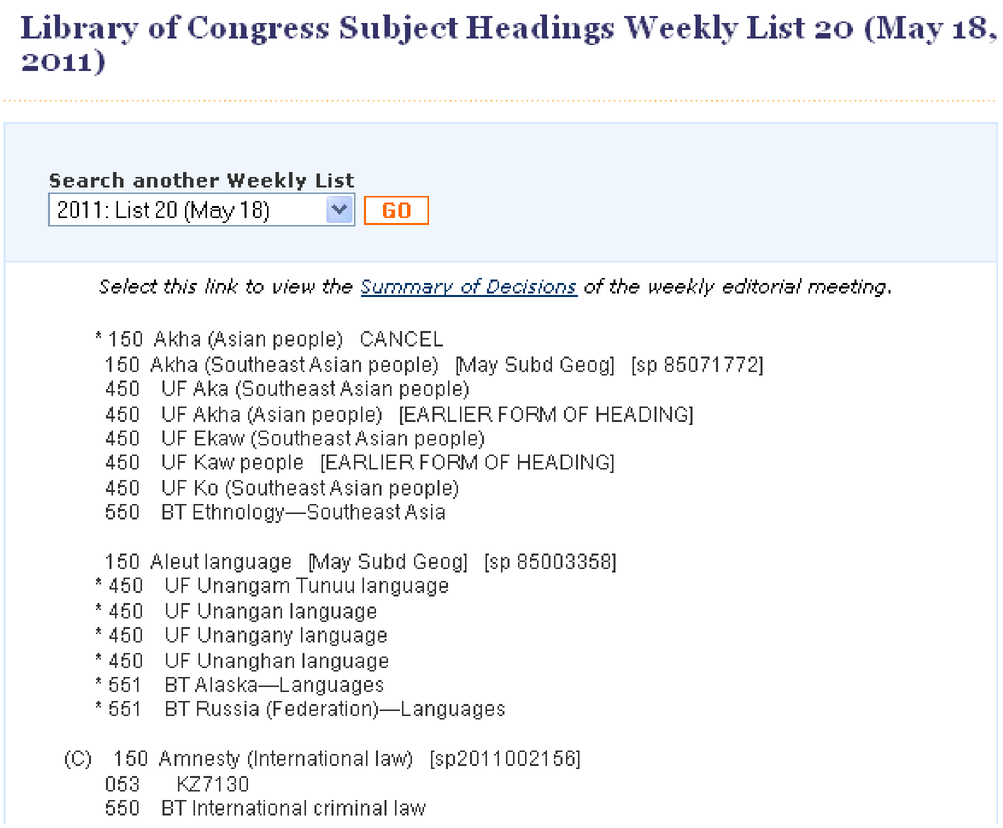
Concept hierarchies are used to organize information since ancient times. A concept hierarchy is a hierarchical organization which provides an overview of data at different levels of granularity. A concept hierarchy includes the *concepts* discussed in a certain domain, and the *relations* among those concepts. Concepts represent the topics of interests in the domain and relations determine how to organize the concepts into a hierarchy.

Concept hierarchy is a good choice to organize data for later access and use. Concept hierarchies can provide an overview of the data and presents a “lay of the land” of the range of the issues raised in the data. It can also provide direct access to information when the concepts are associated with data sources. Data sources can be text documents, videos, audios, blog posts, etc. Concept hierarchies allow a user to quickly access information by "drilling down" into the data sources that discuss particular topics. Examples of concept hierarchies include the Library of Congress Subject Headings[1] (LCSH, Figure 1.1), where the entire Library of Congress are organized into hierarchical structures, the Yahoo! Directory [2] (Figure 1.2) and the Open Directory Project[3] (ODP, Figure 1.3), where the entire Web data are attempted to be organized into hierarchical structures.

Moreover, concept hierarchies are able to pull up and make visible the content which is ranked low in a sequential document list ordered either by creation dates, document docket identification numbers, or relevance to a search query. By pulling up documents that are ranked low, concept hierarchies greatly increase these documents' visibility to the user, hence increase a user's awareness of the overall picture of a domain and makes it easier for the user to directly access particular documents that interest her.

Many concept hierarchies, such as LCSH, Yahoo! Directory and ODP, aim to cover a wide range of topics, support a large set of users and tasks, and usually have long lifespans to serve for a long period of time. The creation and maintenance of these concept hierarchies demand a large amount of manual efforts from multiple experts. Manual construction is often time-consuming; the maintenance of manually-constructed concept hierarchies is often challenging, too. In fact, the high labor cost and the tight time constraints make these concept hierarchies infeasible in many real-life situations, such as when a data analyst must respond in a timely manner and when a Web user needs to organize search results for a short-term information need. Such situations call for task-specific and user-specific concept hierarchies that can be constructed quickly either automatically or semi-automatically. We illustrate two of such situations and show how concept hierarchies can help as follows.

[1]http://www.loc.gov/.
[2]http://dir.yahoo.com/.
[3]http://www.dmoz.org/.


Library of Congress Subject Headings Weekly List 20 (May 18, 2011)

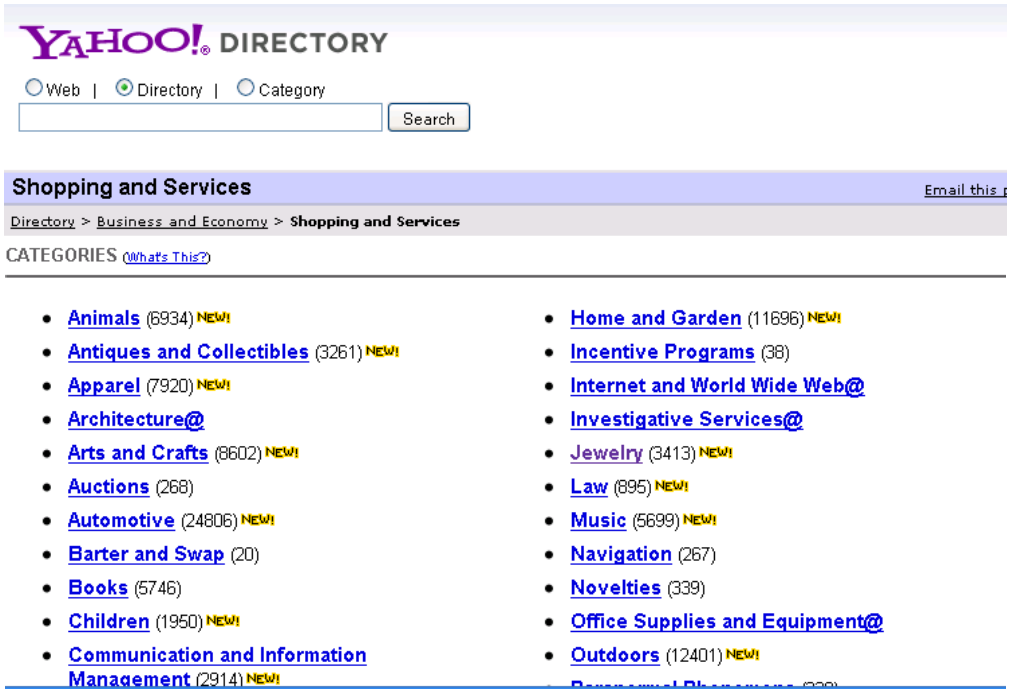
Select this link to view the Summary of Decisions of the weekly editorial meeting.

```
 * 150  Akha (Asian people)   CANCEL
   150  Akha (Southeast Asian people)   [May Subd Geog]   [sp 85071772]
   450    UF Aka (Southeast Asian people)
   450    UF Akha (Asian people)   [EARLIER FORM OF HEADING]
   450    UF Ekaw (Southeast Asian people)
   450    UF Kaw people   [EARLIER FORM OF HEADING]
   450    UF Ko (Southeast Asian people)
   550    BT Ethnology—Southeast Asia

   150  Aleut language   [May Subd Geog]   [sp 85003358]
 * 450    UF Unangam Tunuu language
 * 450    UF Unangan language
 * 450    UF Unangany language
 * 450    UF Unanghan language
 * 551    BT Alaska—Languages
 * 551    BT Russia (Federation)—Languages

(C)   150  Amnesty (International law)   [sp2011002156]
   053      KZ7130
   550    BT International criminal law
```

Figure 1.1: A portion of the Library of Congress Subject Headings (updated list on May 18, 2011).

1.1 Data Exploration in Notice Comment Rulemaking

Jane is a government employee working in the Offce of Policy within the Department of Homeland Security. She participates in the rule-making process in the rules that issued by her department. In Notice Comment Rulemaking [YCS06], rule-makers, including Jane, from administrative agencies of the U. S. government are required to seek comments from the stakeholders and the public, and respond to substantive issues in the final rule within a limited amount of time. By law, rule-makers must consider every substantive issue raised during the comment period. Each year a few high profile rules attract hundreds of thousands of comments. When comment volume is high and the time for processing comments is short, evaluating each comment *quickly* and *thoroughly* brings the rule-makers, such as Jane, a significant burden.



Shopping and Services

Directory > Business and Economy > Shopping and Services

CATEGORIES (What's This?)

- Animals (6934) NEW!
- Antiques and Collectibles (3261) NEW!
- Apparel (7920) NEW!
- Architecture@
- Arts and Crafts (8602) NEW!
- Auctions (268)
- Automotive (24806) NEW!
- Barter and Swap (20)
- Books (5746)
- Children (1950) NEW!
- Communication and Information Management (2914) NEW!
- Home and Garden (11696) NEW!
- Incentive Programs (38)
- Internet and World Wide Web@
- Investigative Services@
- Jewelry (3413) NEW!
- Law (895) NEW!
- Music (5699) NEW!
- Navigation (267)
- Novelties (339)
- Office Supplies and Equipment@
- Outdoors (12401) NEW!

Figure 1.2: A portion of the Yahoo! Directory.

For a public comment set submitted about a rule, Jane would like to have a quick overview of the public comments (about a certain rule) that submitted to her deparment. A concept hierarchy particularly constructed for public comments about a certain proposed rule is desirable for her to have a quick overview of the "lay of the land" of the comments and to well understand the range of the issues raised in the comments. Figure 1.4 shows a hierarchy manually built by government employees using ICF's CommentWorks[4] for the set of public comments about "Minimum Standards for Driver's Licenses and Identification Cards Acceptable by Federal Agencies for Official Purposes" (Docket id: DHS-2006-0030). The dataset contains 210,000 public comments submitted to the U.S. Department of Homeland Security (DHS) during a sixty-day comment period in 2007. For such large amount of email comments sent from all around the country, Jane needs to find out what the major issues are and respond to the issues within a limited period of time.

CommentWorks displays a concept hierarchy on its left pane. The hierarchy shown in Figure 1.4 organizes the comments according to the subjects mentioned in the proposed

[4]http://www.icfi.com/insights/products-and-tools/commentworks/.

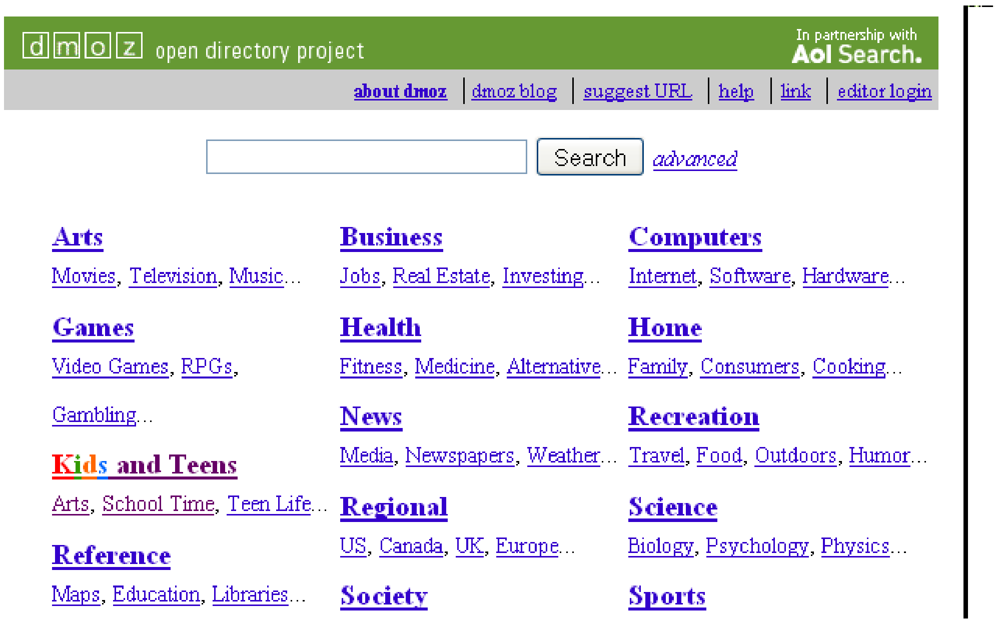
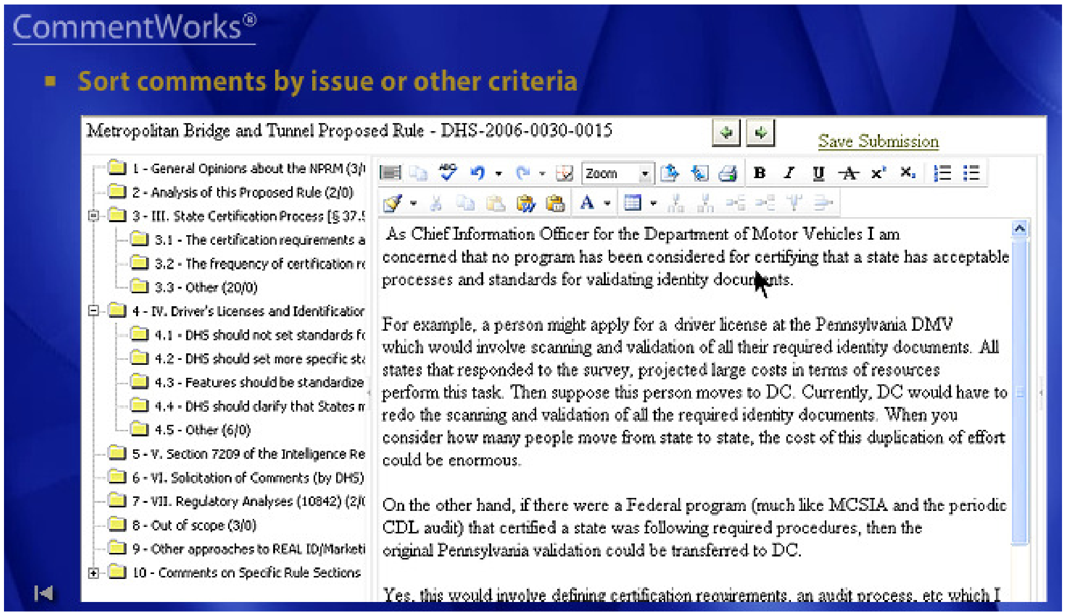

dmoz open directory project
In partnership with Aol Search.
about dmoz | dmoz blog | suggest URL | help | link | editor login
Search advanced
Arts
Movies, Television, Music...
Business
Jobs, Real Estate, Investing...
Computers
Internet, Software, Hardware...
Games
Video Games, RPGs, Gambling...
Health
Fitness, Medicine, Alternative...
Home
Family, Consumers, Cooking...
News
Media, Newspapers, Weather...
Recreation
Travel, Food, Outdoors, Humor...
Kids and Teens
Arts, School Time, Teen Life...
Regional
US, Canada, UK, Europe...
Science
Biology, Psychology, Physics...
Reference
Maps, Education, Libraries...
Society
Sports

Figure 1.3: The top level of the Open Directory Project (ODP).

rule. The top-level concept clusters include *general opinions*, *analysis of the proposed rule*, *state certification process*, *driver's license and identification*, *intelligent records*, *solicitatior of comments*, *regulatory analysis*, and *out of scope*. Once the concept hierarchy is built, Jane can browse the concept hierarchy, click one of the nodes in the hierarchy, and easily access the set of comments related to that concept. She is happy with this organization of the comments.

Mike is another employee working in the same office as Jane's. He helps to write review reports for public comments. He is also involved in the outreach program of the department and often pays attention to comments sent from different groups of stakeholders. Therefore when Mike organizes comments, he often oraganizes them based on the stakeholders who care about this proposed rule. For instances, for the comments in Figure 1.4 Mike organizes them into top-level concept clusters including *drivers*, *government officers*, *immigration lawyers*, and *professors who study political science*. This is a totally different organization from the previous concept hierarchy for the same public comment set organized by Jane.

Jane and Mike organize the same set of public comments differently to serve their own purpose. The multiple concept hierarchy representations enable them organizing the same


CommentWorks®

- Sort comments by issue or other criteria

Metropolitan Bridge and Tunnel Proposed Rule - DHS-2006-0030-0015

Save Submission

- 1 - General Opinions about the NPRM (3/
- 2 - Analysis of this Proposed Rule (2/0)
- 3 - III. State Certification Process [§ 37.5
  - 3.1 - The certification requirements a
  - 3.2 - The frequency of certification r
  - 3.3 - Other (20/0)
- 4 - IV. Driver's Licenses and Identificatior
  - 4.1 - DHS should not set standards f
  - 4.2 - DHS should set more specific st
  - 4.3 - Features should be standardize
  - 4.4 - DHS should clarify that States n
  - 4.5 - Other (6/0)
- 5 - V. Section 7209 of the Intelligence Re
- 6 - VI. Solicitation of Comments (by DHS)
- 7 - VII. Regulatory Analyses (10842) (2/
- 8 - Out of scope (3/0)
- 9 - Other approaches to REAL ID/Marketi
- 10 - Comments on Specific Rule Sections

As Chief Information Officer for the Department of Motor Vehicles I am concerned that no program has been considered for certifying that a state has acceptable processes and standards for validating identity documents.

For example, a person might apply for a driver license at the Pennsylvania DMV which would involve scanning and validation of all their required identity documents. All states that responded to the survey, projected large costs in terms of resources perform this task. Then suppose this person moves to DC. Currently, DC would have to redo the scanning and validation of all the required identity documents. When you consider how many people move from state to state, the cost of this duplication of effort could be enormous.

On the other hand, if there were a Federal program (much like MCSIA and the periodic CDL audit) that certified a state was following required procedures, then the original Pennsylvania validation could be transferred to DC.

Yes, this would involve defining certification requirements, an audit process, etc which I

Figure 1.4: A concept hierarchy for public comment dataset "Metropolitan Bridge and Tunnel Proposed Rule", by CommentWorks.

data into different views due to different preferences and purposes. The resulting hierarchy depends on the preferences of the employee who builds the hierarchy. Each concept hierarchy is a personalized concept hierarchy.

Concept hierarchies can be of little use once the task is done. Jane and Mike are free to keep it for later uses or share with other users with similar information needs. But, they are also free to discard the concept hierarchy as soon as it is not their interests any more. Therefore, this type of concept hierarchies must be built in a relatively cheaper way, either automatically or semi-automatically.

1.2 Search Result Organization in Web Search

In the last two decades, search engines have become the most popular online tools to search for information. Search engines invest most of their resources in *looking up* relevant documents. Many search queries submitted to Web search engines such as Google, Bing or Yahoo! are just revisiting queries, i.e., queries aiming to find pages that have already been visited by a user at an earlier time. In fact, more than 50% Web page visits are re-visits [ATDE09]

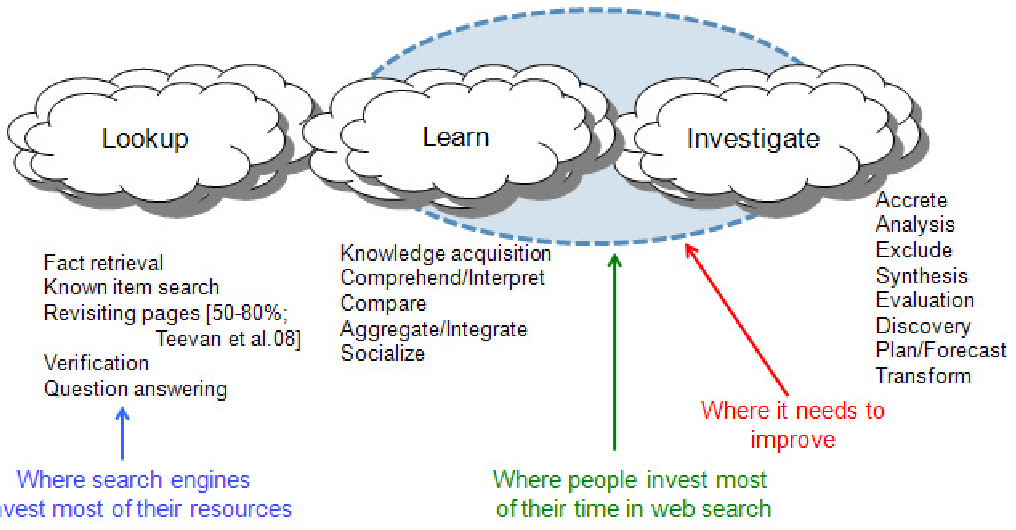


Figure 1.5: Concept hierarchies help in learning and investigation. (Adapted from a slide by Gary Marchionini [Mar07].)

and more than a third of Web searches are re-finding queries [TAJP07]. In these Web page revisits, a user is crystal clear about what the search target is. Usually a search target is just one Web document. Search engines perform pretty well in finding a single search target, and people are satisfied with the search performance for these simple look-up search tasks.

In many other cases, however, the search target is not a singleton, but multiple parts in multiple Web documents. This frequently occurs for *task-based queries* where the information need is *to learn* and *to investigate*. The user has to extracts important concepts that she cares about from multiple search results and organizes the concepts into some structures that make sense to her. She usually does the extraction and organization in her mind. Usually, the information need is too complicated to be fully expressed by a single search query. We therefore often see that a user has to reformulate and resubmit her search queries many times to a search engine in order to find useful information. This process is time-consuming.

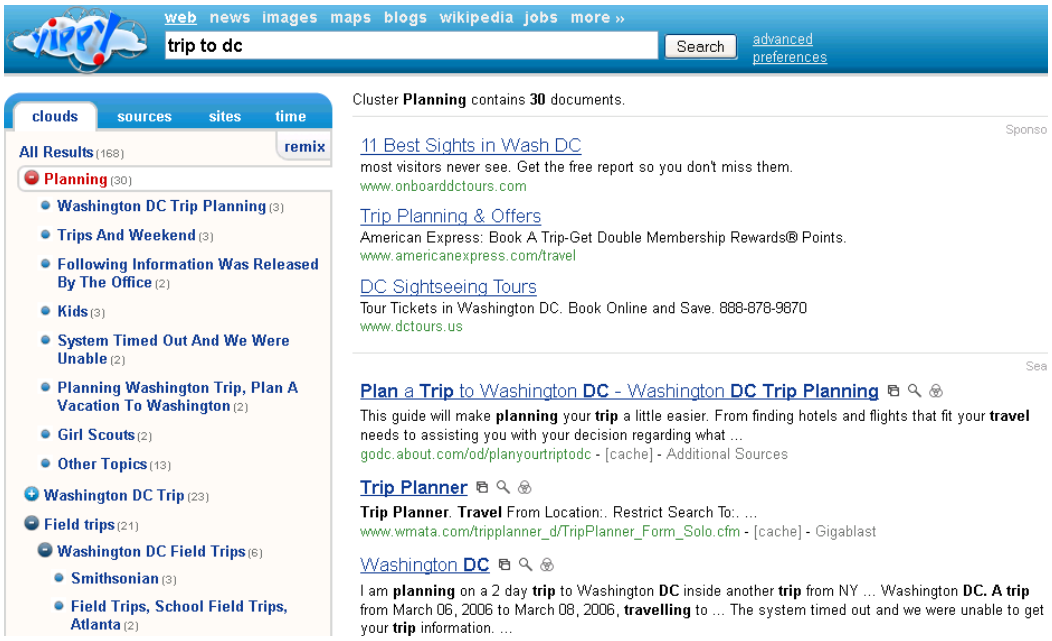
Marchionini pointed out that people actually invest most of their time in Web search in *examining* search results and in *learning* and *investigating* [Mar07]. Such activities include exploring the structure of the retrieved Web documents, acquiring the knowledge, comparing the search results, comprehending the meanings, analyzing or evaluating the materials (Figure 1.5). That long process of examining search results, repeatedly reformulating search queries, and trying different ways to express what is in one's mind is tiresome and frustrating.

For example, Jeniffer is a relative of Jane. She loves to go to fields trips. Jennifer wants to come to Washington, DC to visit Jane and her family. When Jeniffer planning her trip to DC, she submits a search query "trip to DC" to a Web search engine to gather information. The search engine returns a long list of blue links (a list of more than 500,000 document URLs), each of which leads to a Web document. Jeniffer must click some, sometimes many, of these links, read the documents, extract the related information, and organize it by herself. Here, search engines push the burden of processing and organizing information to the user, which is not desirable and clearly needs to be improved.

A concept hierarchy built over the search results is able to show the "landscape" of the topics and hence save Jane from this tiresome information triage process. Figure 1.6 shows a hierarchy constructed by Yippy [5] for search results of the query "trip to DC". The hierarchy is built automatically by clustering the results generated by the search engine. Specifically, the hierarchy groups the search results into *planning*, *Washington DC trip*, *field trips*, and a few more related topics. With the help of this concept hierarchy, Jennifer can easily view the options and make her decision about the trip. Therefore it becomes easier for her to find her path to reach the relevant documents. Concept hierarchies arethus able to go beyond the basic search and provide significant help for the task of information triage/seeking.

Similar to rule-makers' personal preferences in organizing the public comments, individual Web user's preferences also play a large role in how to organize the information. Different people probably plan their trips differently and wish to show their own preferences in the concept hierarchies. Jeniffer may want to build the hierarchy as shown in Figure 1.6, organizing the Web search results into different plans for field trips. Another user may build the hierarchy based on *places of interests* (*museums, historical buildings*) and *transportation options* (*subway, taxi, train*). An ideal concept hierarchy in these situations should be able to capture the user preferences.

[5]http://search.yippy.com/.


web news images maps blogs wikipedia jobs more »
trip to dc
Search
advanced
preferences
clouds sources sites time
remix
Sponso
Sea

All Results (168)
- Planning (30)
  - Washington DC Trip Planning (3)
  - Trips And Weekend (3)
  - Following Information Was Released By The Office (2)
  - Kids (3)
  - System Timed Out And We Were Unable (2)
  - Planning Washington Trip, Plan A Vacation To Washington (2)
  - Girl Scouts (2)
  - Other Topics (13)
- Washington DC Trip (23)
- Field trips (21)
  - Washington DC Field Trips (6)
    - Smithsonian (3)
    - Field Trips, School Field Trips, Atlanta (2)

Cluster **Planning** contains **30** documents.

11 Best Sights in Wash DC
most visitors never see. Get the free report so you don't miss them.
www.onboarddctours.com

Trip Planning & Offers
American Express: Book A Trip-Get Double Membership Rewards® Points.
www.americanexpress.com/travel

DC Sightseeing Tours
Tour Tickets in Washington DC. Book Online and Save. 888-878-9870
www.dctours.us

Plan a **Trip** to Washington **DC** - Washington **DC Trip Planning**
This guide will make **planning** your **trip** a little easier. From finding hotels and flights that fit your **travel** needs to assisting you with your decision regarding what ...
godc.about.com/od/planyourtriptodc - [cache] - Additional Sources

Trip Planner
Trip Planner. **Travel** From Location:. Restrict Search To:. ...
www.wmata.com/tripplanner_d/TripPlanner_Form_Solo.cfm - [cache] - Gigablast

Washington **DC**
I am **planning** on a 2 day **trip** to Washington **DC** inside another **trip** from NY ... Washington **DC. A trip** from March 06, 2006 to March 08, 2006, **travelling** to ... The system timed out and we were unable to get your **trip** information. ...

Figure 1.6: Clustering of Web search results of "Trip to DC", by Yippy.

1.3 Personalization in Concept Hierarchies

We have seen two examples where concept hierarchies, especially personalized concept hierarchies, can help in information triage and access. It is unlikely that such concept hierarchies can be directly made of or adapted from existing manually-crafted concept hierarchies, such as ODP, Yahoo! Directory, and LCSH. The following summarizes the issues that potentially make existing concept hierarchies fail in learning and in investigating a specific new domain.

First, there may simply be no existing concept hierarchy that can provide the right coverage of concepts and details for a task. For example, public comments are collected from the general public about a proposed rule that often does not exist before. No concept hierarchy would possibly exist before the rule is issued and the comments are collected. Even if a rule is related to some existing concept hierarchies, for example, the aforementioned rule by DHS may be related to a portion of ODP about *driver's license*, the relation is limited. Most existing concept hierarchies are built beforehand by domain experts and nearly impossible to contain the specific concepts in the rule and in the comments.

Second, even if existing concept hierarchies include the concepts that the user is interested

The most agreeable concept pairs		**The least agreeable concept pairs**	
Information			
Concept pairs (parent,child)	*Agreements*	*Concept pairs (parent,child)*	*Agreements*
telecommunications telecommunications re-sellers	21	information, sound recording studios	1
publishing industries, software publishers	20	newspapers & publishers, book publishers	1
publishing industries, greeting card publishers	19	sound recording industries, radio stations	1
telecommunications, wired telecommunications carriers	19	publishers, music publishers	1
publishing industries, newspaper periodical book and directory publishers	19	production, sound recording studios	1
Kindergarten			
program, summer program	24	example of schools, waldorf school	1
program, early childhood program	24	staff, person	1
skill, cognitive skill	23	school topics, art	1
program, back-up care program	23	development stages of child, toddler	1
student, lower school student	23	subjects, communication	1
Polar bear			
hunter, sport hunter	24	laws, polar bear recovery plan	1
pollution, greenhouse gas pollution	23	polar bear protection act, polar bear recovery plan	1
energy, traditional energy	23	organization, gas industry	1
hunter, trophy hunter	23	stakeholder, pollution producer	1
extinction, mass extinction	22	organization involved, polar bear specialist group	1

Table 1.1: The most and the least agreeable concept pairs for datasets *information*, *kindergarten*, and *polar bear*.

in, there are many ways to organize a given set of concepts. For example, *jewelries* can be organized by *types of gemstones*, or by *brands*. Existing concept hierarchies are mainly created by domain experts, who are probably not the users. Thus a user can only consume the provided concept hierarchy and cannot input her opinion to customize the hierarchy in any way. The pre-determined, static organization is not tailored to specific tasks or for specific users. In the situations where user preferences play a significant role (e.g., trip planning), a static organization of information is not flexible enough to serve the purpose.

In most cases, just like personal computers and search engines, concept hierarchies only work for one user. We admit that some differences between concept hierarchies constructed by different people are caused by multiple facets or mixed initiatives that can be agreed and shared by multiple users. However, for a concept hierarchy construction tool to support a single user, it needs to adapt to *the* user's personal preferences. Whether her preference comes from different facets or mixed-initiatives, or purely comes from her unique point of view, makes little difference in terms of training the tool. Because for this tool, unless collaboration among multiple users has to be taken into account, all its training and learning data is supplied by this user only. Therefore, in this dissertation research, we focus on studying concept hierarchy construction with personal preferences.

1.3.1 An Experiment on Personal Differences in Concept Hierarchies

To better understand personalization in concept hierarchies, we explore the concept hierarchies constructed by real users in a user study (more details about the user study are shown in Chapter 6 and Chapter 7). In particular, we look for commonality and differences among the concept hierarchies constructed by the participants.

Twenty-four participants were involved in the user study, hence we have twenty-four concept hierarchies constructed for each of the 20 datasets. The datasets cover a wide range of domains, such as "organizing information-related terms", "planning a trip to DC", "finding a good wedding videographer", and "organizing financial terms" (more details in Chapter 3 and Chapter 6). To study the commonality and the differences between concept hierarchies, we break each concept hierarchy into pairs of parent and child nodes, and count how many participants agree on a pair. The agreements range from 1 to 24.

Table 1.1 lists the most agreeable and the least agreeable concepts by the participants. Here we only show three representative datasets: the"information" dataset, the "kindergarten" dataset, and the "polar bear" dataset. The table lists the 5 most agreeable and the 5 least agreeable pairs of parent and child concepts in the concept hierarchies as well as how many participants agree on organizing them in that way.

We find that there exist concepts that all participants agree on how to organize them and there also exist concepts that no participant agrees on how to organize them. And this is true for every dataset that we examine. For instance, in "kindergarten" dataset, "program" is the parent concept for "early childhood program", which all 24 participants agree. These most agreeable concept pairs show that people can indeed agree on how to organize certain concepts. However, only a few pairs have more than 5 people to agree on how to organize them.

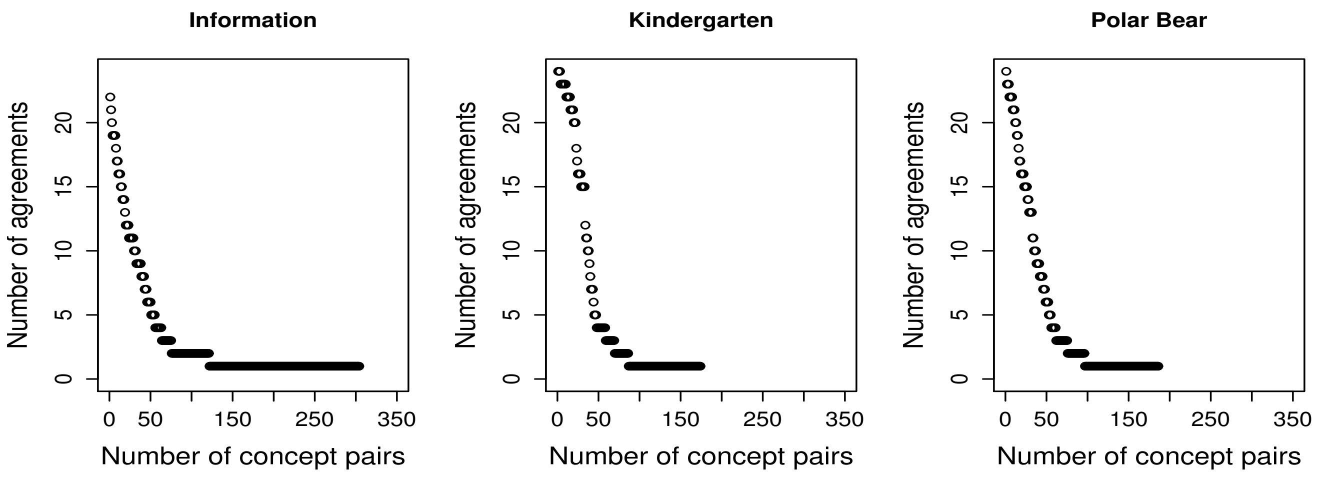


Figure 1.7: Agreements among participants for the parent-child pairs (for three example concept hierarchies: *information*, *kindergarten*, and *polar bear*).

The unique concept pairs that no participant agrees on, i.e., concept pairs having only 1 participant votes for them, show personal preference among the participants. For example, one participant considered "publishers" is the parent node of "music publishers", while other participants did not organize them in that way. There are much more pairs of nodes that are uniquely organized by a participant than pairs of nodes that are commonly agreed by many participants.

We further plot the number of agreements for every concept pair in the three example datasets in Figure 1.7. We observe a long-tail power-law distribution in the plots for all three datasets. In particular, we find that in the dataset "information", there are about 300 unique concept pairs, while in the datasets "kindergarten" and "polar bear", more than 200 unique concept pairs exist.

This shows that although commonality and differences co-exist in concept hierarchies created for the same dataset by different participants, the differences are much more dominate than the commonality. People use rich and diverse expressions to construct concept hierarchies and organize information differently within them. The differences between concept hierarchies constructed by different people are considered as *personalization in concept hierarchies.*

Personal Concept Hierarchy Construction is the outcome when concept hierarchy construction meets with personal preferences. 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 concept hierarchy construction* in this dissertation.

In the remainder of this chapter, we present the challenges, approach, and contributions of this dissertation research, and outline the structure of this document.

1.4 Challenges

The challenges of personal concept hierarchy construction rise from both *concept hierarchy construction* and *personalization*.

A major challenge in *concept hierarchy construction* is to extend the existing work on concept hierarchy construction as well as to present new solutions. Existing work on concept hierarchy construction has been conducted under a variety of names, such as, ontology learning, taxonomy induction, semantic class learning, relation acquisition, and relation extraction. The existing approaches fall into two main categories: *pattern-based* and *clustering-based*. *Pattern-based* approaches define lexico-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 or represented by patterns. The direct implication of this limitation is that only a small number of relations are captured. *Clustering-based* approaches hierarchically cluster terms based on their semantic similarities. In order to derive the semantic similarity between terms, terms are usually represented by a vector of features in the semantic space. These approaches complement the 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 largely due to inaccurate estimations of semantic similarities among concepts. Concept hierarchy construction demands for new solutions that extend the existing technologies and combine the strengths of both approaches naturally and flexibly into a unified framework. With this new framework, we are able to not only greatly improve the accuracy of concept hierarchy construction, but also investigate and evaluate the impact of the individual techniques in existing approaches.

Another challenge in *concept hierarchy construction* is to deal with *concept abstractness.* Concepts can be divided into abstract concepts and concrete concepts. 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. In a hierarchy, concrete concepts usually lay at the bottom of the hierarchy while abstract concepts often occupy the intermediate and the top levels. The obvious differences between the two types of concepts suggest that there is a need to treat them differently in concept hierarchy construction. 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 similarly hoping that the impact of concept abstractness on concept hierarchy construction is small and different behaviors of concrete and abstract concepts can be captured by lexico-syntactic patterns. In this dissertation research, we take the challenge and propose to explicitly model concept abstractness in concept hierarchy construction.

A third challenge in *concept hierarchy construction* is to deal with *concept coherence.* Sometimes, concepts along a branch in a hierarchy may not be coherent. This problem is mainly caused by *polysemy* - multiple meanings for the same word. For example, two parent-child relations, “financial institutions $\rightarrow$ bank” and “bank $\rightarrow$ Monongahela River”, without special constraints are connected to form a longer concept chain “financial institutions $\rightarrow$ bank $\rightarrow$ Monongahela River”. This concept chain is obviously invalid at the semantic level. The challenge is how to introduce proper constraints to guarantee that polysemies go to different branches and concepts within the same branch are coherent. In this dissertation, we show how to enforce concept coherence in long distance relations as one of the optimization criteria in the proposed framework.

A fourth challenge in *concept hierarchy construction* is to *fairly evaluate the quality of a concept hierarchy.* This problem can be transformed into a task of measuring the similarity between a constructed hierarchy and a reference hierarchy. The more similar the automatically constructed hierarchy is to the reference hierarchy, the better the quality the constructed hierarchy is. Although it is generally accepted that similarity measure for hierarchies is the Tree Edit Distance [Bil05][ZSS92], its NP-completeness and MAX SNP-hard property [ZJ94] make it infeasible to be widely used in real applications. To the best of our knowledge, no standard feasible method is available for measuring hierarchy similarity. A new solution is needed for hierarchy similarity measurement.

The challenges of *personalization* are also significant. The first challenge in *personalization* is how to *incorporate personal preferences* in the concept hierarchy construction process. When assisting one to organize information, personal concept hierarchy construction must adapt to her personal understanding of the problem, to her preference towards a certain aspect of the problem, and to her 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 both are bears. Neither way is wrong; the choice is simply due to different personal criteria. Personal preferences need to be captured during concept hierarchy construction and to be reflected in a general human-teaching-machine-learning procedure.

Moreover, as a practical system, the second challenge in *personalization* is to *respond in real-time*. Personal concept hierarchy construction needs to interact with a user and collaboratively construct the hierarchy. This requires an interactive learning algorithm to quickly adjust and make predictions based on a few user inputs as training data. In real-time interactions, a learning algorithm needs to be efficient enough to quickly customize the formulas and statistical learning models. Since we use trees, whose computation and construction could be expensive [ZJ94], it is crucial to find good constraints to greatly reduce the search space in order to respond in real time.

Last but not least, *understanding user behaviors* for personal concept hierarchy construction and studying the possible implications of the behaviors are also challenging. Although the focus of this dissertation is on how to build light-weight concept hierarchies that reflect personal preferences, it is also important to investigate whether, how, and why concept hierarchies constructed by different individuals are different. It is interesting to understand the underlying reasons of people's preferences for concept hierarchies, whether people are self-consistent, whether their preferences for concept hierarchies are caused by their different construction methods (manual or interactive), use of different semantic feature functions, or differences in demographics (such as gender, major, and age).

With all the challenges in mind, and all the possibilities that these challenges can bring to us, we explore the wonderful field of *personal concept hierarchy construction*.

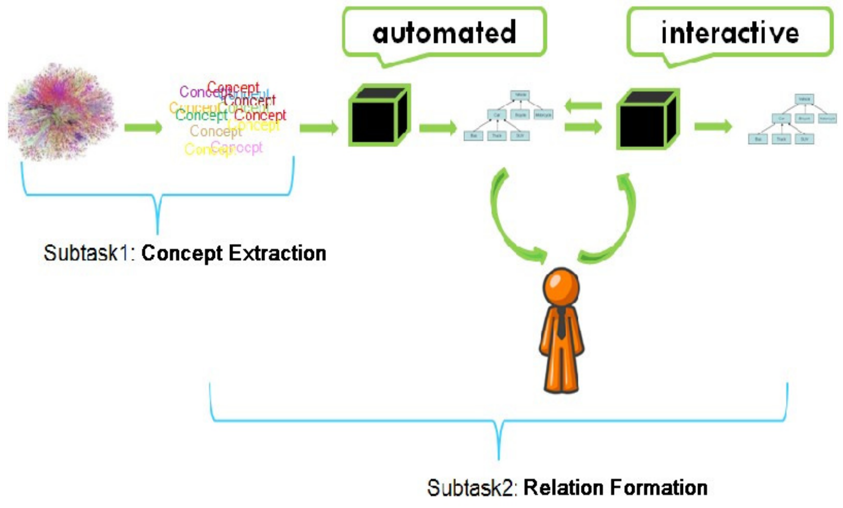


Figure 1.8: Constructing a personal concept hierarchy.

1.5 Our Approach

Personal concept hierarchy construction concerns task specifications and user preferences. With a large set of unstructured data, our goal is to organize the relevant information within a domain into an easy-to-comprehend concept hierarchy that suits specific needs for both the user and the task. Figure 1.8 demonstrates the proposed process of how to construct a personal concept hierarchy. Personal concept hierarchy construction consists of two subtasks: *concept extraction* and *relation formation*.

Concept extraction acquires concepts from a given dataset such as a document collection. Concepts are topics of interest in the given dataset. They usually are nouns, noun phrases, or named entities. They are directly extracted from the document collection, whose size can range from a few hundreds to millions of documents. Techniques of how to extract concepts are presented in Chapter 4 - *concept extraction*.

After the concepts are extracted, their relations need to be identified. The relations determine how the concepts are organized and how the resulting concept hierarchy look like. Relation formation in this dissertation research is done by going through two processes. The first is an fully-automated process, which proposes an initial concept hierarchy to the user so that it saves the user's effort from building everything from scratch. It incrementally clusters concepts based on semantic distances between concepts and transforms the task of concept hierarchy construction into a multi-criterion optimization based on optimization of hierarchy structures and modeling of concept abstractness and concept coherence. This automatic framework is presented in Chapter 5 - *metric-based concept hierarchy construction*.

Once an initial concept hierarchy is presented to a user, she works interactively with the system to construct a personal concept hierarchy. This interactive process adopts a human-guided machine learning approach. In this interactive process, the user interacts with the system and makes improvements to the concept hierarchy. The system learns what the user changes and adapts to make sensible predictions about the remaining un-organized concepts and provides its suggestions to the user. After observing what the system suggests, the user evaluates them and makes a few more improvements if necessary. Through several iterations of exchanging opinions between the user and the system, a personal concept hierarchy that satisfies the user's need is finally constructed. This interactive framework is presented in Chapter 6 - *human-guided concept hierarchy construction*.

1.6 Contributions of This Dissertation

This dissertation presents new techniques for personal concept hierarchy construction. In particular, it elaborates both automatic concept hierarchy construction as well as interactive concept hierarchy construction. Our approach combines the strengths of existing pattern-based and clustering-based approaches by employing a feature representation which includes heterogeneous features. The features vary from simple statistics to complicated syntactic dependency features, basic word length to comprehensive Web-based contextual features. The flexible design of the learning framework allows us to use all but not limited to these features. This more general feature representation has the potential to learn more complex concept hierarchies than prior studies. 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 *metric-based* concept hierarchy construction framework, presented in Chapter 5, addresses concept abstractness in concept hierarchy construction and enforces concept coherence in long distance relations. It models abstract concepts and concrete concepts differently in the learning framework to produce more sensible results. The incremental clustering process transforms concept hierarchy construction into an optimization problem based on three desirable properties: *minimum evolution*, *concept abstractness*, and *concept coherence*. An evaluation with WordNet and Open Directory Project data demonstrates that the framework is effective for concept hierarchy construction.

The *human-guided* concept hierarchy construction framework, presented in Chapter 6, addresses personalization and human computer interaction in the process of personal concept hierarchy construction. Periodic manual guidance provides training data for learning a distance metric, which is then used during automatic activities to further construct the concept hierarchy. A user study demonstrates that human-guided machine learning is able to generate concept hierarchies with manually-built quality. The *human-guided* concept hierarchy construction framework is the first concept hierarchy construction framework to construct personalized and task-specific concept hierarchies.

This dissertation also studies user behaviors during concept hierarchy construction. It explores whether people create concept hierarchies more quickly or more consistently using the proposed methods, whether there are consistent dataset-specific or user-specific differences in the concept hierarchies that people construct, whether people are self-consistent, and how these factors interact with different construction methods. The user study elaborates that human-guided concept hierarchy construction is promising since it not only reduces time and effort as expected but also provides assistance when a user knows little about a domain.

In addition, this dissertation contributes to hierarchy similarity measurement. We present a unique bag-of-word representation for hierarchies and a novel similarity metric, Fragment-Based Similarity (FBS), for measuring hierarchy similarity. We discuss and empirically evaluate various design decisions that lead to the proposed similarity measure. Most importantly, we propose a very efficient similarity measure which well approximates Tree Edit Distance with a time complexity of only $O(n^3)$.

The research in this dissection is the first step of personal concept hierarchy construction, and an important step forward of concept hierarchy construction. It develops both automated and interactive methods that assist information seeking, organization, and management activities, especially methods that will not only lead to practical systems of immediate benefit, but also push our ability to reason about the sophisticated information systems of the future.

This work addresses important problems of personal concept hierarchy construction, 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 constructing personal concept hierarchies.

1.7 Outline

The rest of this dissertation is organized as follows. Chapter 2 discusses the related work. Chapter 3 presents the problem, datasets, software tool, and evaluation metrics used in this dissertation research. Chapter 4 presents the techniques for concept extraction. Chapter 5 describes the fully-automatic metric-based concept hierarchy construction framework. Chapter 6 presents the human-guided concept hierarchy construction framework. Chapter 7 elaborates the user study. Chapter 8 summarizes the main ideas of this dissertation, addresses the contributions of this dissertation research, and describes some concrete issues as the future work.

Chapter 2

Related Work

This chapter reviews the related work to this dissertation research. The related work is about three aspects of concept hierarchy construction, namely ontology learning, human-guided machine learning, and interactive techniques for concept hierarchy construction.

2.1 Ontology Learning

There has been a substantial amount of research on 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 two most popular classes of approaches are *pattern-based ontology learning* and *clustering-based ontology learning.*

2.1.1 Pattern-Based Ontology Learning

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

For example, one pattern for the most widely used *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 many instances for *is-a* relation. Each instance contains a pair of noun phrases, filling the positions of *NPx* and *NPy*. For example, a noun phrase pair *NPx* = *'lawn spray'* and *NPy* = *'chemicals'* can be found in the sentence "Mercury, lawn spray, and other chemicals are harmful to fetus development". The pattern identifies an instance of *is-a* relation: *is-a(lawn spray, chemicals)*.

Once an instance of relation is identified, it can be used to find more patterns and instances through an iterative technique called *bootstrapping*. Bootstrapping is commonly used in pattern-based approaches [Hea92, ECD$^+$05, GBM03, RH02, PP06]. It utilizes a few handcrafted 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. For example, if "lawn spray" and "chemicals" both appear in a sentence "Commonly-used chemicals include lawn spray and mercury", we may infer *NPy include NPx* as a new pattern for *is-a* relation. Bootstrapping is effective and scalable to large datasets; however, uncontrolled bootstrapping soon generates undesired instances once a noisy pattern is brought into the cycle.

Pattern-based approaches have been applied to extract various types of semantic relations, including *is-a*, *part-of*, *sibling*, and many others. We review a few approaches for each type of relation as follows.

Is-A Patterns

Pattern-based approaches started from and still pay a great deal of attention to the most common *is-a* relation. It is also known as hypernym relation. Hearst pioneered using a handcrafted list of *is-a* patterns, such as "*NPx, and/or other NPy*" and "*NPy including NPx*", as seeds and bootstrapping to discover new instances and patterns for *is-a* relation [Hea92]. In her work, Hearst manually identified six commonly used lexico-syntactic patterns that indicated *is-a* relation and used them as the seed patterns. The seed patterns were used to search for instances of *is-a* relation in text. Each identified instance, consisting of a few noun phrases, was used to search contexts where the noun phrases occurred syntactically near one another. The contexts were recorded and the common contexts (selected manually) yielded new patterns that indicated *is-a* relation. The new patterns were then used to discover more instances of the relation. Through this bootstrapping technique, abundant new instances of *is-a* relation were identified; these instances were successfully used to verify and augment the WordNet [Fel98] noun ontologies.

Since Hearst's work, many approaches have used lexico-syntactic patterns in their work on *is-a* relation. For instance, Mann built a fine-grained proper noun ontology from news texts [Man02]. Mann extracted instances of *is-a* relation for proper nouns using a single pattern: "*(the)? NPy NPx*", where *NPy* is a noun phrase tagged with NN/NNS (noun or plural noun) and *NPx* is a proper noun phrase tagged with NNP/NNPS (proper noun or plural proper noun). Similar to Hearst patterns, *NPy* represents the parent and *NPx* represents the child. For example, one instance of this pattern is *the automaker Mercedes-Benz*, from which we can get a relation *is-a(Mercedes-Benz,automaker)*. Mann claimed that "*(the)? NPy NPx*" is more productive than Hearst patterns for the corpus he used. Mann also used the acquired instances to augment WordNet and to infer answers for TREC-style factoid questions [Voo02]. He further extended the pattern's coverage by inferring new instances through a simple rule. The rule was "*is-a(NPa, NPy) and is-a(NPb, NPy) and is-a(NPb, NPz)* $\Longrightarrow$ *is-a(NPa, NPz)*". For example, if we knew instances *is-a(Mercedes-Benz, automaker) and is-a(Opel, automaker) and is-a(Opel, car company)*, we could infer a new instance, *is-a(Mercedes-Benz, car company)*. Such inference rule was able to extend the coverage of lexico-syntactic patterns, however, as pointed out by Mann himself, care must be taken to ensure the inferences were proper. However, Mann did not discuss how to ensure proper inferencing in his paper.

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

Etzioni et al. bootstrapped lexico-syntactic patterns and discovered named entities at the Web scale in the KnowItAll system [ECD$^+$05]. The KnowItAll system is an earlier version of the TextRunner system [BE08]. This work looked for class members of a certain category, for instances, names of scientists. We can view categories as parent concepts, and class members as child concepts. The authors used three methods to extract the class members. The first was a set of domain-specific extraction rules based on linguistic analysis. The second was lexico-syntactic patterns, such as "chemist" and "biologist" are class members of "scientists". The third method was an HTML-based list extractor to extract class members as entries in a list on Web pages. It learned a wrapper for each list and then extracted name entities from the list. KnowItAll was probably the first Web-scale fact extractor in literature. It extracted over 50,000 named entities. A challenge with such Web scale knowledge acquisition system was how to increase the recall without sacrificing the precision. KnowItAll was one of the earliest attempts of using statistical measures, in particular point-wise mutual information (PMI), to select and control the quality of the extracted named entities.

Kozareva, Riloff, and Hovy proposed a single double-anchored pattern, "*NPy such as NPx and NPz*", for recognizing child concepts for a given category (parent) [KRH08]. The first noun phrase *NPy* indicates the parent, the second noun phrase *NPx* indicates a child, the thrid noun phrase *NPz* indicates another child. This pattern was used to find instances for *is-a* relations, particularly to find more children for a known parent. The double-anchored pattern was more specific since both the parent and a child are required to appear in the pattern. Not surprisingly, the double-anchored pattern is more accurate with the sacrifice of pattern coverage. The pattern could also bootstrap to get more instances and then substituted the new instances as the child in another double-anchored pattern. The bootstrapping increased coverage, but also introduced many incorrect instances. To control the quality of the instances, the authors employed a graph representation of concept linkages, and then measured the candidate instances' popularity (a candidate was discovered by other instances) and productivity (a candidate lead to discoveries of many other instances). Based on these two measures, the quality of new instances were well-controlled.

Sibling Patterns

Another commonly-used relation is *sibling*, which describes the relation of sharing similar meanings and being members of the same class.

An observation is that sibling words are often co-occur together in text. For example, they often co-occur in conjunctions (*lions, tigers, and bears*), lists (*lions, tigers, bears ...*), appositive (*the stallion, a white Arabian*), and nominal compounds (*Arabian stallion; tuna fish*) [RS97]. A conjunction of noun phrases are phrases connected by *and*, for instance, *vice-president and his wife*, *evaluation methodology, tools and equipment*. An appositive is a phrase connected by comma and referring to the same concepts. For example, *Steve Jobs, president of Apple Inc.*. Inspired by the observation, Riloff and Shepherd used co-occurrence statistics in local context to discover instances of *sibling* relation for nouns [RS97]. Their work used a small set of seed words that belonged to a given category (a parent concept), and identified more words that also belonged to that category. 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 showed high conditional probability within the context windows were selected as new seed words. The authors reported that an eight-iteration bootstrapping worked well. Although this work did not directly use lexico-syntactic patterns to mine instances, it used narrow context windows to mimic patterns which represent conjunctions, lists, appositive, and nominal compounds. Therefore, this work is still considered as a pattern-based approach. A limitation of this work is that it only handled nouns, not noun phrases.

Roark and Charniak also used co-occurrence statistics in local context to discover instances of *sibling* relation [RC98]. They extended Riloff and Shepherd's work to noun phrases by only counting head noun's co-occurrence within a sentence. A head noun in a noun phrase is the noun that determines the semantics of the phrase; other words in the noun phrase modify the head noun. For example, in the sentence "*A cargo aircraft, fighter plane, or combat helicopter ...*", only *aircraft*, *helicopter*, and *plane* would be counted as co-occurring with each other. In contrast, in Riloff and Shepherd's work, *cargo*, *aircraft*, and *fighter* would also be counted as co-occurring with each other; the calculation was incorrect in this case. Moreover, Roark and Charniak tried to distinguish nominal compounds whose non-head noun was a legitimate member of a class versus nominal compounds whose non-head noun was 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 relative frequency of a noun occurs as a non-head noun in any nominal compound over this noun occurs in any nominal compound. If the relative frequency was above the threshold, this non-head noun was kept as a new child for the parent indicated by the head noun. For instance, in *fighter plane* the non-head noun *fighter* was a child of *plane*; whereas in *government plane* the non-head noun *government* was not a child of *plane* because *government* was usually not a non-head noun. The selected patterns are similar to Mann's within-phrase pattern "*(the)? NPy NPx*" [Man02].

Widdows and Dorow built a graph representation of similar words to extract the sibling concepts [WD02]. Only one symmetric sibling pattern, "*NPx and/or NPy*" was used in their work. It was similar to using conjunctions. Each pair of concepts matched with this pattern were considered as sibling concepts and between them potentially existed a link in a graph where semantically-related words were connected. For example, "apple" could be related to "pear", "ibm", "tree", and "novell" in the graph. However, not every single instance matched the above pattern was added into the graph. A new node to the graph was the node that was neighbor to the most existing nodes in the graph. Ten word categories were studied in the work. They are *crimes, places, tools, vehicle conveyance, musical instruments, clothes, diseases, body parts, academic subjects*, and *food stuffs*. One contribution of this work was that the graph built upon noun co-occurrence flow naturally to recognize polysemies. Multiple word sense ambiguity was captured by different siblings to a node in the graph.

Davidov and Rappoport also used symmetric patterns for identifying sibling relations [DR06]. They proposed an interesting approach of discovering new patterns. There were two steps in this approach. First, meta patterns were used to identify pattern candidates. The meta patterns were composed as "CHC", "CHCH", "CHHC", and "HCHC". "C" meant a content word, which contained real semantic meanings. "H" meant a high frequency word, which usually contained no actual meaning. For example, "apple" and "student" were content words, "is" and "or" were high frequency words. Such general meta patterns could identify many candidate patterns. Second, among those candidates, symmetric patterns were searched in order to identify sibling words. Similar to [WD02], symmetric patterns were collected through a graph representation of the concepts. The graph was a uni-directional graph with the content words being the nodes and the high frequency words being the arcs. Content word could be the slots for sibling words. The arc could represent a pattern between two slots. A pattern that allowed a large percentage of content words to appear at both ends (not just one of them) of an arc was considered as a good pattern, because it indicated symmetric relations. If the graph was further constrained to be bidirectional, the rate between the number of symmetric nodes in the bidirectional graph and that in the uni-directional graph could be used to filter out the low quality patterns. The higher the rate, the better a pattern. The high quality symmetric patterns were then used to extract sibling words.

Part-of Patterns

The third common type of relation is *part-of*. Part-of relation is also known as "part-whole" relation or "meronym" relation. Part-of relation is a bit more complicated than *is-a* relation and *sibling* relation because there are many different kinds of part-whole structures. As pointed out by Girju et al. in [GBM03], there exist six types of part-whole relations: *component-integral* (wheel-car), *member-collection* (student-university), *portion-mass* (meter-kilometer), *stuff-object* (alcohol-wine), *feature-activity* (paying-shopping), and *place-area* (DC-USA).

The earliest research on automatically identifying *part-of* relations was Berland and Charniak's work [BC99]. In their work, they only used two *part-of* patterns, "*NPy's NPx*" and "*NPx of a/an NPy*", to discover instances with *part-of* relation. NPy indicates the *whole* and NPx indicates the *part*. The example instances for these two patters included "the basement of the building" and "the building's basement" [BC99]. No bootstrapping was used in their work to discover more patterns or more instances as usually pattern-based approaches do. The authors used two statistical metrics to rank and select the matched instances. The first metric was a log-likelihood metric which measured how surprised one would be to observe the difference between the frequency of a word indicating the whole and the frequency of a word indicating the part. The second metric measured how far apart the distributions of *p(whole|part)* and *p(whole)* were if a significant level of 0.05 or 0.01 was required. The authors claimed that the second metric was better since it did not overestimate the importance of data frequency at the expense of the difference between *p(whole|part)* and *p(whole)*.

Girju et al. took a bootstrapping approach similar to [Hea92] for *part-of* relations. Although they pointed out six types of *part-of* relations, the authors did not distinguish the differences between different types of relations in their paper. The discovered patterns were either phrase-level patterns in examples such as “high heel shoes”, “girl’s mouth”, “eyes of a baby”, and “door knob”, or sentence-level patterns in examples such as “the wheel is part of the car” and“ the car contains four wheels” [GBM03]. The selection of new patterns and new instances from the discovered ones was done by a supervised decision tree learning algorithm.

Patterns for Other Types of Relations

Other types of relations that have been studied by pattern-based approaches include synonyms and antonyms by Lin et al. [LZQZ03], verb relations (including *similarity*, *strength*, *antonym*, *enablement* and *temporal*) by Chklovski and Pantel [CP04], question-answer relations (such as *birthdates*, *why-famous*, and *inventor*) by Ravichandran and Hovy [RH02], general purpose analogy by Turney et al. [TLBS03], entailment by Szpektor et al. [STDC04], and more specific relations, such as *consist-of*, *purpose*, *creation*, and *completion* by Cimiano and Wenderoth [CW07], *LivesIn(Person, Location)*, *EmployedBy(Person, Organization)*, and *CorpAcquired(Organization, Organization)* by Bunescu and Mooney [BM07]. Due to space limitation, we focus on three representative works and present them below.

Lin et al. studied how to identify synonyms and antonyms in [LZQZ03]. The widely-used Distributional Hypothesis states that words with similar meanings tend to appear in similar context [Har54]. However, Lin et al. pointed out that many closely located words were actually not synonyms, but antonyms. For example, “ally” was often closely located to “adversary”, but it was an antonym not a synonym to “adversary” [LZQZ03]. The authors took a clever two-step approach to remove antonyms in the synonym candidate pool suggested by Distributional Hypothesis. First, two patterns *from X to Y* and *either X or Y* were used to identify antonyms. The ratio of the number of words matched in these two patterns and the number of words appearing in AltaVista search results was used to decide whether two words were indeed antonyms, which were then removed from the synonym candidate pool. Second, a bilingual dictionary was used to check whether two words’ translations in another language were the same. For example, “advocate” and “attorney” could both be translated to the French word *defenseur* [LZQZ03]. Therefore, they remained as synonyms in the pool.

Chklovski and Pantel used 35 lexico-syntactic patterns to identify five verb relations [CP04]. The five verb relations were *similarity* (maximize-enhance, produce-create), *strength* (taint-poison, permit-authorize), *antonyms* (buy-sell, live-die), *enablement* (access-review, accomplish-complete), and *happens-before* (marry-divorce, tie-untie) [CP04]. The authors used word co-occurrence in the Web to find highly-related verbs. Each pair of such verbs, based on their specific pattern type, were assigned a semantic relativeness score by human assessors. For example, for a strength verb relation, a judge was asked to evaluate whether *outrage* was stronger than *shock* by judging whether the pattern "X outrage Y" was stronger than the pattern "X shock Y". The work was among one of the first for identifying verb relations by using lexico-syntactic patterns.

Ravichandran and Hovy employed machine learning techniques to learn lexico-syntactic patterns for question answering [RH02]. It has been noted that in the task of question answering, certain types of questions and answers were expressed by fixed patterns. For example, for a *birthdate* question "when was X born?", the answers to it typically fell into one of the following patterns: "X was born in YEAR", "X (YEAR-YEAR)", and "X (YEAR -)". The answer patterns consisted of specific types of named entities or parts-of-speech, and could be expressed as lexico-syntactic patterns. The authors used the AltaVista search engine to get the text pool and bootstrapped more answer instances and more answer patterns from the pool. Specifically, a question's key terms and a known answer were submitted together to AltaVista. The top 100 returned documents were split into sentences. Only the sentences that contained both the question terms and the answer terms were used to infer more patterns. The authors successfully applied this technique to find good patterns for several question types, including *birthdate*, *why-famous*, and *inventor*.

Advantages

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

Bootstrapping is widely used in pattern-based approaches. It is able to scale to text collections as large as the Web. It is a data-driven approach and helps to find more patterns which may not be easily hand-crafted by a human expert. However, it needs to control the quality of the new patterns acquired. 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 modeling for multiple membership [ZZSW09].

Weaknesses

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, a corpus' size is often not that large therefore the problem still exists. Moreover, since patterns usually extract instances in pairs, e.g., a pair of parent-child or a pair of siblings, the approaches suffer from the problem of inconsistent concept chains after connecting pairs of instances to form concept hierarchies. For instance, *car* has hyponyms of *coupe*, *sedan* and *sport*, *sport* has *basketball*. It is obvious that *car* should not be an ascendent of *basketball* in this context.

2.1.2 Clustering-Based Ontology Learning

Clustering-based ontology learning approaches apply the techniques in hierarchical clustering [BPd$^+$92, Lin98] to organize concepts into ontologies. They usually represent word contexts as numeric vectors and cluster words based on similarities of those vectors. Since clustering is often based on similarity of concepts, clustering-based approaches usually only deal with *is-a* and *sibling* relations.

Based on the types of clustering algorithms used, clustering-based ontology learning can be further divided into three groups: agglomerative approaches, divisive approaches, and incremental approaches.

Agglomerative-Clustering-Based Ontology Learning

Agglomerative clustering, also known as bottom-up clustering, iteratively merges the most similar clusters into bigger clusters [BPd$^+$92, Car99]. In agglomerative-clustering-based ontology learning, an algorithm organizes the concept hierarchy in a bottom-up fashion. It first groups the most similar concepts together, usually two by two, to form clusters at the leaf level. It then names the newly-formed clusters and moves up in the hierarchy to merge the most similar clusters iteratively until all concepts are merged into one big cluster.

Lin defined a similarity measure between two words based on their occurrences in syntactic dependency triples [Lin98]. A syntactic dependency triple consisted of two words and their grammatical relationship. For example, Lin defined the triples "brown adj-mod-of dog", "have subject I", "I subject-of have", "dog obj-of have", "dog adj-mod brown", "dog det a", and "a det-of dog" for the sentence "I have a brown dog". The triples becomes patterns when some slots in the triples were substituted with wildcards. The frequency of every pattern was counted. The similarity of two words was defined over the number of various patterns with wildcards containing one or both of them. Particularly, point-wise mutual information (PMI), cosine, Hindle, Dice, and Jacard were used to calculated the similarity. PMI was reported that it gave the best performance. The authors then used an agglomerative clustering algorithm to group concepts as well as to compare a concept group to existing ontologies such as WordNet.

Caraballo represented a word's context as a vector containing the counts of every other word appearing in conjunction or in appositive with the word in a corpus [Car99]. All words were then agglomeratively clustered based on the cosine similarity between the vectors. The approach took a rather standard agglomerative clustering approach to group similar concepts in a bottom-up fashion. When combining the two most similar words into a new common parent, the new parent node's similarity was computed over an weighted average of the similarities between each of its children and other existing nodes in the ontology. Note that the new parent node had no name, just represented by the set of the child nodes.

Rosenfeld and Feldman used surface text patterns as features, selected the features by statistical measures, and then applied agglomerative clustering based on cosine similarity between the feature vectors. Instead of directly using lexico-syntactic patterns as features, the authors used some arbitrary surface text patterns. They were further filtered by how often a word matched a pattern in text. The authors found that among all the possible patterns, only those sufficiently powerful contextual patterns produced good results. By implementing and comparing several agglomerative options, the authors reported that the most effective option was the single linkage method.

Divisive-Clustering-Based Ontology Learning

Another type of clustering algorithm is divisive clustering. Divisive clustering, a.k.a. top-down clustering, iteratively splits clusters into smaller clusters using a partitioning clustering algorithm, such as K-means [Mac67]. This procedure is applied recursively until each concept has its own singleton cluster.

Clustering By Committee (CBC) is an example of divisive-clustering-based ontology learning [PL02, PR04]. The CBC algorithm first found the most similar words among all pairs of words, then identified several sets of highly-related words. These highly-related words were named as "committees". The committees formed the core of each cluster. The cluster centroids were obtained by averaging the feature vectors of the committees. The algorithm then iteratively assigned the remaining words, those did not belong to any committee, into the existing clusters represented by committees.

Cimiano et al. used bi-secting k-means divisive clustering for ontology learning [CHS04]. The algorithm initiated with a randomly selected word, and then computed the centroid of the entire dataset and the difference between the centroid and the random word. It then divided the data space into two subclusters, according to the relative distances of all the words in the two subclusters to the current cluster centroids. The cluster centroids were then updated based on the feature vectors of existing words in the two subclusters. Note that the features were semantic features which represented by a feature lattice. The updates of clustering centroid and the cluster membership alternatively iterated until the clustering was stable.

Similar to agglomerative clustering approaches, divisive clustering approaches create new non-leaf nodes during the clustering process. For an ontology, these new non-leaf nodes need to be named. However, naming clusters is a very challenging task.

Incremental-Clustering-Based Ontology Learning

Both agglomerative and divisive clustering approaches face the challenge of how to appropriately label non-leaf clusters. In agglomerative clustering, clusters are merged into bigger clusters, which need to be labelled; in divisive clustering, parents of split clusters also need to be labelled. Labelling amplifies the difficulty in creation and evaluation of ontologies.

The incremental clustering approaches avoid this problem by assuming that all the concepts are known and make-do on the available concepts as the best choices. It adds concepts and relations into an ontology one at a time. All nodes, including both leaf and non-leaf nodes, are assumed coming from the existing set of concepts. The advantage of the incremental approach is that it eliminates the trouble of inventing cluster labels and concentrates on placing concepts in the correct positions in an ontology.

The work by Snow et al. [SJN06] took an incremental approach of ontology construction. In their work, an ontology grew based on maximization of conditional probability of relations given evidence. The evidence was the instances matched with patterns defined on syntactic dependency parse trees. It is the most similar piece of work to our approach, however there are significant difference from their work to this dissertation research. The concept hierarchy construction algorithm presented in this dissertation research grows a concept hierarchy based on optimization of hierarchy structures and modeling of concept abstractness and concept coherence (Chapter 5) while Snow et al. used maximization of conditional probability. Moreover, our approach employs heterogeneous features from a wide range while Snow et al. used only syntactic dependency. We compare system performance between this work and the metric-based concept hierarchy construction framework in Chapter 5.

Advantages

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 can be identified. Clustering-based approaches also avoid the problem of inconsistent chains by addressing the structure of a concept hierarchy globally from the outset.

Weaknesses

Nevertheless, it was pointed out by researchers that clustering-based approaches cannot generate relations as accurately as pattern-based approaches. As reported by Pantel and Pennacchiotti, clustering-based approaches generally failed 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 [PTL93], syntactic dependency [PR04, SJN05], surface patterns [RF07], co-occurrence [YC08a], conjunction and appositive features [Car99]. So far, there is no systematic study of which features are the best for automatic concept hierarchy construction under various conditions. This dissertation research is the first to study how various types of features interact with different types of relations for concept hierarchy construction (Chapter 5).

2.1.3 Other Ontology Learning Approaches

Besides the popular pattern-based and clustering-based approaches, researchers proposed other interesting approaches for concept hierarchy construction. We present several approaches in this subsection.

Sanderson and Croft identified *is-a* relation by studying the ratio of document frequencies for two concepts [SC99]. In their approach, a concept x was another concept y's parent in a concept hierarchy if the conditional probability $p(x|y)$ is greater than 0.8 and the conditional probability $p(y|x)$ is less than 1. The intuition was that a word x subsumed (i.e. is the parent of) another word y if the documents containing y were a subset of the documents containing x. The threshold of 0.8, instead of 1, was empirically set to relax the restriction a bit to include the cases when y (the child) did not co-occur with x (the parent). The technique worked well for high quality text such as news articles, but did not work well for low quality text such as Web search results. The authors used search query terms and their expanded query terms as the concepts to work on. Those search queries were from the TREC text retrieval tasks [Voo01]. An initial set of documents was retrieved and analyzed to generate concepts co-occurring frequently in different passages. Sanderson and Croft presented several resulting ontologies in their paper. In general, the ontologies looked good. However, some

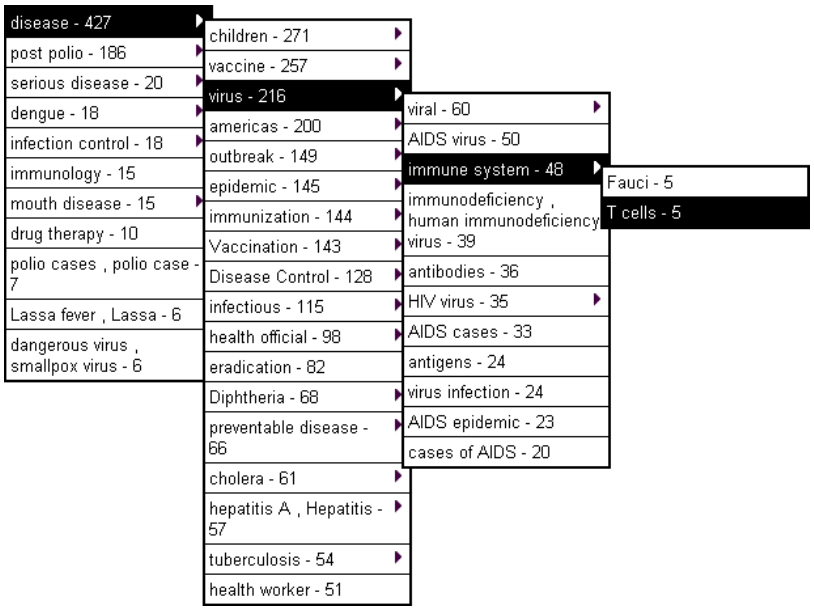


Figure 2.1: A screen capture of Sanderson and Croft's system [SC99].

fragments of the ontologies were not clear why they were organized in that way. For example, an example in [SC99] (Figure 2.1), "disease"$->$"children", "disease"$->$"americas" were two parent-child relations in the hierarchy. It meant "children" and "americas" should be siblings; this seems wrong. This pure statistical approach did not constrain coherence among the concepts, therefore it suffered from the problem of concept incoherence as we mentioned earlier in Chapter 1.

Lawrie and Croft [LC03] employed language models to develop ontologies as document summaries. In particular, they estimated whether a word should be a concept by measuring the Kullback-Leibler (KL) contribution of this word to a set of documents. A concept hierarchy was generated iteratively in a top-down manner. Initially, the document set was the entire collection. Once the words about a topic were found necessary for a topic, they were selected as the concepts at the top level. Then the document collection was reduced to the retrieval results for individual concepts. The child of a concept were determined from this reduced set of documents by the above method again. The approach did not use many features. Instead, it used only one feature - the existence within sub-topics - to derive the organization of concepts. The authors claimed that using only a single feature could produce ontologies easier to interpret.

Cimiano et al. applied Formal Concept Analysis [CHS04] to construct ontologies. They represented concepts as objects and feature lattices, then looked for common features shared by different objects as the super-concepts for them. This approach could be considered similar to the idea of Semantic Web. It required that the semantic information was built into the documents before a concept hierarchy was constructed. Because of this special requirement, it built ontologies easy to understand but was not efficient enough to be applied to large scale datasets.

2.2 Human-Guided Machine Learning

Human-guided machine learning is an emerging topic which unites the research areas of human computer interaction (HCI) and machine learning (ML). It is the study of designing and developing machine learning algorithms which 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 manual guidance on the learning process.

While significant attention has been paid to developing machine learning algorithms trained by data that is provided all at once and 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 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 the most related to our task.

Kerne et al. employed document clustering for generating spacial hypertext documents [KKSM05]. The clustering was done based on meta data, document locations, and user-specified weights to each semantic features in the vectors representing each concept. The most interesting part of this work was that a user could view the relative weights for the semantic features through sliding the boundaries within a pie chart. This allowed the user to directly participate in feature selection and interaction with the clustering algorithm.

Huang and Mitchell defined a two-way-communication between the human and the machine in a mixed-initiative clustering approach [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 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.

In our work, we use human-guided machine learning to collect personal preferences. In Chapter 6, we develop a human-guided concept hierarchy construction framework, which collects manual guidance during the concept hierarchy construction process. The manual guidance is implicitly presented by a partial human-constructed concept hierarchy. The learning algorithm takes the partial human-constructed concept hierarchy 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.

Human-guided machine learning is related to but different from Active Learning [CGJ96, SC00]. Active learning algorithms try to make use of abundant unlabeled data and *actively* requests manual labels for data with low confidence. Lots of effort in active learning has been put on how to select the right data samples for users to label. Human-guided machine learning and active learning are similar in terms of the human involvement in machine learning. 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 while human-guided machine learning aims to design learning algorithms which follow human directions in real-time and produce results that reach human satisfaction.

2.3 Interactive Technologies for Ontologies

Because concept hierarchy construction involves human-computer interaction, it is necessary to review the related research in Human-Computer Interaction (HCI) for interactive technologies related to concept hierarchy construction.

Faaborg and Lieberman presented a web browser which generalized and predicted users' search activities by Programming-By-Example and using knowledge from ontologies represented as networks [FL06]. Examples of ontologies they used included MIT's ConceptNet[1]and Stanford's TAP[2]. The generalization of information was done by matching terms on a page to other terms in ontologies by *is-a* relations. This was basically a rule-based match. By predicts a user's goal, the web browser was able to activate the specific web functions for that goal. This evaluation was done over 34 participants, which is comparable to our user study (24 participants) in terms of scales.

Clark et al. described a graphical interface for users to easily enter knowledge into a knowledge-base, a.k.a. an ontology [CTB$^+$01]. Axioms and facts were displayed as graphical elements, and the participants could create and modify the graph based on the knowledge needed to be entered. The four operations presented in their work were specialize (click a graph node and change the label), add (click an empty button and enter the label), unify (drag and drop a node), and connect (sketch an arc between two nodes). The work considered knowledge capture as a task of graphical component assembly rather than axiom-writing. The evaluation was carried out by only 4 participants working on a moderate size concept hierarchy, which was about a 11-page subsection from a graduate-level textbook on cell biology. One independent biologist was hired as the judge to evaluate the ontologies created by the 4 participants. The biologist rated their work as mostly correct.

Tribble and Rose compared three forms of representations of ontologies, namely graphical network-like representation, hierarchical representation, and list representation of knowledge axioms, for the task of concept hierarchy construction [TR06]. In their objective experimental results, hierarchical representation showed the quickest response time, and the highest correctness for a given task. In their subjective evaluation, hierarchical representation was selected by the participants as the easiest form of representation to understand, as well as

[1]http://web.media.mit.edu/ hugo/conceptnet/.
[2]http://ksl.stanford.edu/projects/TAP/.

the form generating moderate confidence about the answers. Although the scale of the study was not big (only 9 participants), and the statistical analysis revealed no significant effects, hierarchical representation of concept hierarchy clearly showed the potential to be the best way to represent concepts and relations in ontologies.

Ziegler et al. presented a new method of visualizing ontological data [ZKB02]. Basically, it used trees to represent hierarchical structures, such as *part-of* and *is-a* relations. The innovation of this paper was about how to visualize other more specific relations. The proposed method connected the nodes in two trees by an adjacent matrix. In the matrix, nodes of two ontologies were shown along the horizontal and vertical axes of the matrix. The entries in the matrix were marked to indicate whether two nodes that belong to the row and the column were connected with an arc in a relational graph. The user study showed that this matrix browser allowed less search time and better search accuracy in comparison to conventional network-like representations. This evaluation was done over 5 participants only, which is much smaller than our user study in terms of scales.

2.4 Summary

This chapter reviews the related work to this dissertation research from three aspects: ontology learning, human-guided machine learning, and interactive techniques for concept hierarchy construction. Particularly, it extensively reviews various approaches of pattern-based ontology learning and clustering-based ontology learning, as well as emerging approaches in human-guided (mixed-initiative) clustering and different interface designs for interactive concept hierarchy construction.

This chapter provides the necessary backgrounds for the related research fields. In the following chapters, we will mainly present and discuss our own approaches.

Chapter 3

The Problem

Concept hierarchy construction aims to organize information as well as to customize the organization according to user preference. To summarize, the most common situation where people need to use concept hierarchies is when dealing with a large amount of unstructured data, which needs to get sorted, digested, and organized into information content with structure. This structure is adapted to the preference of the person who creates and uses the concept hierarchy.

Before we dive into the specific techniques and algorithms, this chapter presents the background knowledge for this dissertation research. It describes the problem definition, the concept hierarchy construction tool, the datasets and the evaluation metrics used in the dissertation research.

3.1 Problem Definition

As in the examples shown in Chapter 1, Jane works in a government regulatory agency. Her job is to handle public comments sent from the general public about the proposed rules in her agency. For a set of public comments about a certain proposed rule, by law, she must read all comments in a week or two, and provides analysis and generates a summary report to address the important issues and unique opinions raised in the comment set. The issues mentioned in her report will be addressed in the final rule, which will take effect as a federal regulation to influence many families in the country. However, sometimes the amount of comments people submitted could reach hundreds of thousands, which makes it impossible for Jane to handle the public comments and generate her report on time. Jane needs a tool to help her quickly and comprehensively find all the important issues mentioned in the public comments.

In her spare time, Jane likes to travel with her family. Jane is planning a Christmas trip with her kids to Orlando, Florida. She uses a search engine (Google) to gather information. Jane would like to book flight tickets, hotel rooms, car rentals, and theme park tickets. She also needs to plan the multi-day trip to various different theme parks including Disneyland, Universal Studio, Kennedy Space Center, etc. During an economic crisis, Jane also needs to keep the budget affordable hence she wants more information about available promotions and sales. Jane's information need is complex and hence she has to read through many search results returned by the search engine to discover information items that interest her, and probably takes notes or bookmarks the web pages that are useful to her. Moreover, to fulfil such a complex task-oriented search, multiple search queries are issued. Thus Jane has to read even more Web documents to accomplish her task. Again, she needs a tool to help her quickly and comprehensively find all the related information mentioned in the piles of search results.

Many similar examples exist in both our work places and daily life. The common task shared by all these scenarios is that given a document set (could be in any domain), a user wants to be able to quickly overview the important issues within the document set and be able to browse the document set to discover useful information. A concept hierarchy constructed for browsing the document set is necessary to serve this purpose. It can be constructed by extracting the important concepts within the document set and then organizing them into a *semantically meaningful* hierarchical structure to ease information seeking and access within the document set. A concept hierarchy benefits a user's information seeking and access in the following ways:

1. Quickly providing an overview of all important issues in a document set;

2. Producing meaningful and personalized organization of the document set;

3. Increasing the visibility of documents that are ranked low in a list;

4. Presenting the subset of documents about a concept together to allow focused reading for the subset of documents.

Note that concept hierarchy construction is task-specific and user-specific. Whether or not a concept represents an important issue and whether or not the concept hierarchy is semantically meaningful are totally up to the user who creates and uses this concept hierarchy. Therefore, personal preferences from the user are an important factor in constructing concept hierarchy.

Formally, concept hierarchy construction is for a user P to create a concept hierarchy T for a set of documents (or we call it a domain) D. User P provides her manual guidance G to create the personal concept hierarchy T, which contains a set of concepts C for D, as well as the relations R for those concepts.

Specifically, we define that a *personal concept hierarchy* is a data model T that represents a set of concepts C and a set of relations R between those concepts according to manual guidance G for a given domain D.

$$T = (C, R|D, G), \tag{3.1}$$

where $C = \{c_1, c_2, ..., c_n\}, R = \{r(c_1, c_2), r(c_1, c_3), ..., r(c_{n-1}, c_n)\}$, n is the number of concepts, and r is the type of the relation (for example, r can be "is-a", "part-of", "sibling", etc.). The relations are pairwise relations indicating an edge between two concepts in a concept hierarchy. When there is no manual guidance provided, concept hierarchy construction reduces to the task of ordinary automatic concept hierarchy construction, where an *concept hierarchy* can be defined as:

$$T = (C, R|D). \tag{3.2}$$

As we introduced in Chapter 1, concept hierarchies in this dissertation research are considered as tree structures. Every node in the tree corresponds a concept $c \in C$. A concept c is linked to a related subset of documents $d_{c1}, d_{c2}, ... \subseteq D$, such that D is hierarchically organized and user P can browse D via the concept hierarchy T. A document could be associated with multiple concepts.

As shown in Figure 1.8, the process of concept hierarchy construction contains the following basic components: concepts extraction to acquire C, an automatic concept hierarchy construction algorithm to obtain an initial concept hierarchy $T = (C, R|D)$, and an interactive concept hierarchy construction algorithm to obtain the personal concept hierarchy $T = (C, R|D, G)$. These algorithms are presented in the following chapters.

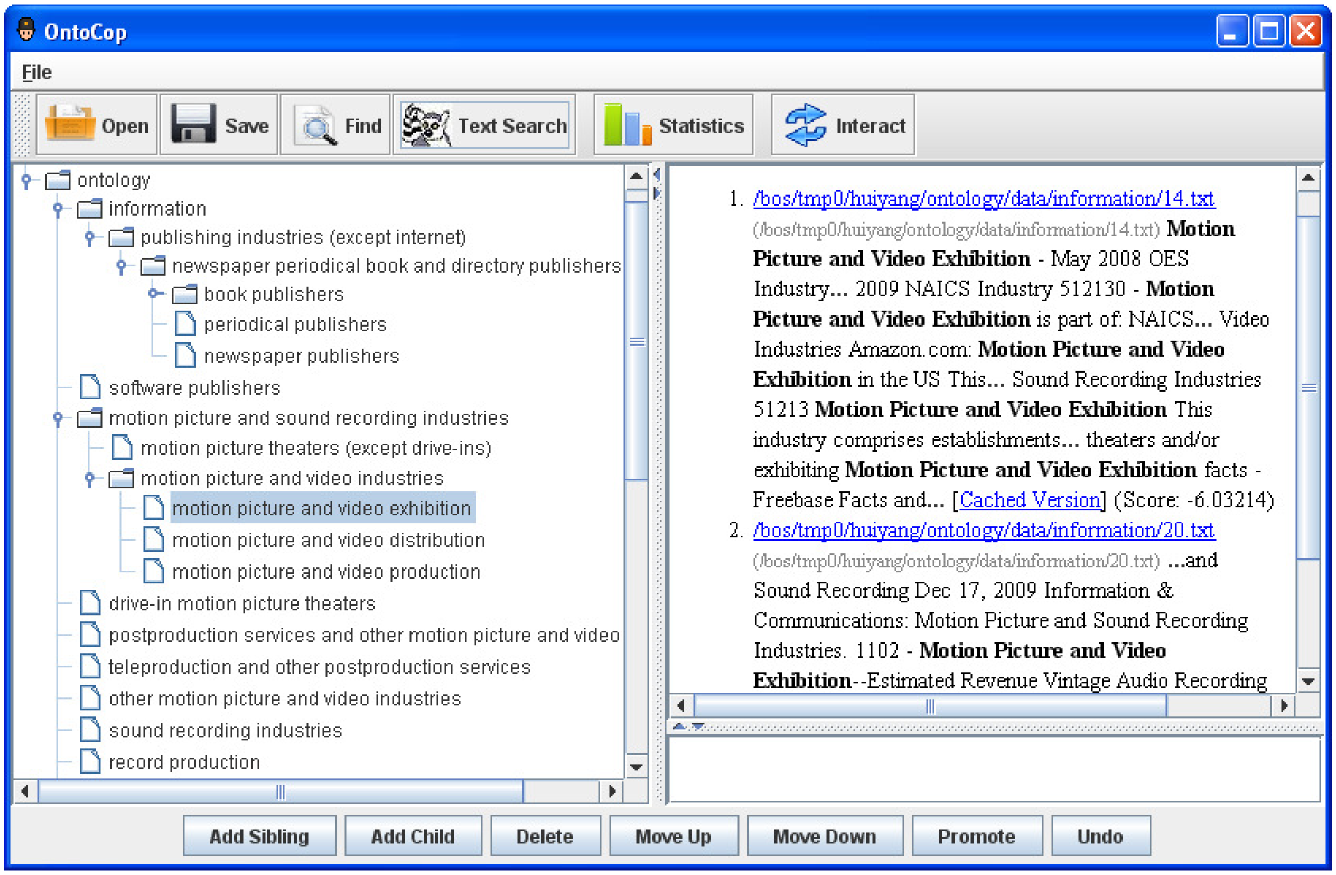
3.2 OntoCop - A Concept Hierarchy Construction Tool

Concept hierarchy construction studies how to organize the information in the documents into a concept hierarchy reflecting user preference. In this process, a user needs a software tool to construct concept hierarchies, which incorporates the techniques presented in this dissertation research. Our tool is called OntoCop (**Onto**logy **Co**nstruction **P**anel) [YC08a, YC08b]. It is a Java application developed as part of this dissertation research. It can support both manual and interactive methods to construct concept hierarchies.

Figure 3.1 shows the user interface of OntoCop. The left pane displays a concept hierarchy as a tree. Each node in the tree represents a concept in the concept hierarchy. The hierarchical organization of the nodes indicates the hierarchical relations of the concepts. Particularly in this example, the concept hierarchy deals with the “is-a” relation among the concepts. Therefore a node on an upper level is the parent of the nodes under it. The user can select a node and move it around by dragging and dropping it in the tree. This is the major functionality that a user needs to use in constructing a concept hierarchy. She can also change the name of a concept by double-clicking a node and entering a new name for the node.

The lower toolbar displays buttons for other concept hierarchy editing functions. The user can use “add sibling” and “add child” buttons to add a new node to the concept hierarchy as the sibling or the child of an existing node. She can delete a node by selecting the node and clicking the “delete” button. She can also move a concept up and down or promote it to the upper level by selecting the node and clicking the “move up”, “move down”, or “promote” buttons. The user can also undo her actions by clicking the “undo” button one or more times.

For manual concept hierarchy construction,a user only employs the above editing functions. She is able to provide the organization of concepts with the help of our editing functions such as dragging & dropping and renaming.



ontology
information
publishing industries (except internet)
newspaper periodical book and directory publishers
book publishers
periodical publishers
newspaper publishers
software publishers
motion picture and sound recording industries
motion picture theaters (except drive-ins)
motion picture and video industries
motion picture and video exhibition
motion picture and video distribution
motion picture and video production
drive-in motion picture theaters
postproduction services and other motion picture and video
teleproduction and other postproduction services
other motion picture and video industries
sound recording industries
record production

1. /bos/tmp0/huiyang/ontology/data/information/14.txt
(/bos/tmp0/huiyang/ontology/data/information/14.txt) **Motion Picture and Video Exhibition** - May 2008 OES Industry... 2009 NAICS Industry 512130 - **Motion Picture and Video Exhibition** is part of: NAICS... Video Industries Amazon.com: **Motion Picture and Video Exhibition** in the US This... Sound Recording Industries 51213 **Motion Picture and Video Exhibition** This industry comprises establishments... theaters and/or exhibiting **Motion Picture and Video Exhibition** facts - Freebase Facts and... [Cached Version] (Score: -6.03214)
2. /bos/tmp0/huiyang/ontology/data/information/20.txt
(/bos/tmp0/huiyang/ontology/data/information/20.txt) ...and Sound Recording Dec 17, 2009 Information & Communications: Motion Picture and Sound Recording Industries. 1102 - **Motion Picture and Video Exhibition**--Estimated Revenue Vintage Audio Recording

Figure 3.1: A screen capture of OntoCop.

The right pane of OntoCop displays the related documents for a selected concept in the concept hierarchy. When the user wants to know more about a concept and its context, she can first select the concept and then click the “text search” button on the upper toolbar to retrieve the documents that mention the concept from the indexed document collection. OntoCop provides a link and a summary for each relevant document, and highlights the concepts in the summaries. The user views the summaries or reads the documents if she wants to know more about the selected concept. This functionality helps her to better understand the meaning of a concept in the specific corpus and to better construct the concept hierarchy.

The upper toolbar displays buttons for the non-editing functions. The user can use the buttons to “open” a concept hierarchy from an existing file, “save” a concept hierarchy to a file with the extension ".ont", and "find" a particular concept in the concept hierarchy. She can perform "text search" for the related documents for a concept as we described earlier. She can also view the statistics of the current concept hierarchy by using the "statistics" function, which displays the number of nodes in the hierarchy, the depth of the hierarchy, and how many nodes locate at each levels of the hierarchy.

The last button on the upper toolbar is the "interact" button. It is related to interactive method of concept hierarchy construction. We describe the "interact" function more in Chapter 6.

3.3 Datasets

We expect that OntoCop and the techniques presented in this dissertation research are general enough to be applied on various domains. We hence examine our system on datasets from a variety of domains. The datasets can be categorized into two different types of datasets: document collections and reference concept hierarchies.

The "document collection" datasets contain no inherent concept sets and hence the concepts must be extracted from the documents. This type of datasets includes the public comment datasets and the Web datasets. They are both text, but they are different kinds of text. The public comments datasets cover email communications about a specific federal policy or rule. These datasets are from the public; each has a narrow domain. The Web datasets cover search results for specific Web queries that submitted to search engines. These datasets contain properly edited Web pages.

The "reference concept hierarchy" datasets have well-defined concept sets and relations that represent a community agreement about how to organize information; they provide gold standards, but no documents. This type of datasets include hierarchy fragments extracted from the North American Industry Classification System (NAICS), WordNet, and Open Directory Project (ODP). They are used to test to a community-agreed gold standard how close a constructed hierarchical organization can be. NAICS datasets cover broad and diverse concepts and are used to test agreement with industrial standards. WordNet is an English dictionary and is used to test general purpose hierarchical organization for words. ODP is used to test agreement with community-agreed Web document classifications.

Statistic	**TRI**	**Wolf**	**Polar Bear**	**Mercury**	**Trans. Fee**	**Organic**
#comments	86,763	282,992	624,947	536,975	100,732	207,936
#unique comments	16,367	59,109	73,989	104,146	60,345	67,525
#words	1,862,180	6,404,810	13,110,343	10,456,425	5,625,534	6,546,732
#words after duplicate removal	141,063	707,515	472,366	6,327,563	764,347	775,334
#vocabulary	6,582	27,742	16,346	31,975	25,556	30,876

Table 3.1: Statistics of public comment datasets.

3.3.1 The Public Comment Datasets

The U.S. law defines a process known as *Notice and Comment Rulemaking* that requires regulatory agencies to seek comments from the public before establishing new regulations. Most proposed regulations attract few comments from the public, but each year a few regulations attract 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 images to text, only text emails are used in the experiments reported in this dissertation.

Public comments usually include many comments that are generated based on form letters written by special interest groups, such as *BushGreenWatch*, which claimed to "track the Bush administration's environmental misdeeds". Modifying an electronic form letter is very easy, hence many people have begun doing so to better express 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 near-duplicate documents as well as the duplicated texts in documents. After duplicate detection, there are still tens of thousands of unique comments.

Six public comment datasets are used for the experiments in this dissertation, namely "Toxic Release Inventory (TRI)" (Docket id: USEPA-TRI-2005-0073),"Wolf" (USFWS-R6-ES-2008-0008), "Polar Bear" (USDOI-FWS-2007-0008), "Mercury" (USEPA-OAR-2002-0056), "Transportation Registration Fee (Transportation Fee)" (USDOT-PHMSA-2006-25589), and "National Organic Program (Organic)" (USDA-TMD-94-00-2).

The TRI dataset was collected 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 Toxic Release Inventory to reduce reporting burden while continuing to provide valuable information to the public.

The Wolf dataset was solicited by the U.S. Department of Interior, Fish and Wildlife Service (FWS) in 2008. These public comments are about a rule that the FWS 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 dataset was also gathered by FWS in 2007. These public comments are about a rule proposed to list the polar bears as a threatened species under the Endangered Species Act and to initiate a scientific review to study the current situation and future of polar bears.

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

The Transportation Fee dataset was sought by the U.S. Department of Transportation (DOT) in 2006. The dataset is about a rule proposed to increase the registration fees for persons who provide services to transport hazardous materials within the country and outside the country.

The Organic dataset was gathered by the U.S. Department of Agriculture in 2003. It is about a proposed rule of establishing the National Organic Program. The program was proposed for establishment of national standards governing the marketing of certain agricultural products as organically produced to facilitate commerce in fresh and processed food and to assure consumers that such products meet consistent standards.

Table 3.1 displays the total 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, and the vocabulary size for the public comment datasets.

Among the six public comment datasets, the TRI dataset is used to train the participants for the user study described in Chapter 7. Other five public comment datasets are used as testing tasks in the user study.

Table 3.2: Statistics of the Web Datasets

Datasets	# documents	# words	# vocabulary
Find a good kindergarten	100	17,050	3,767
Purchase a used car	100	23,840	2,911
Plan a trip to DC	100	18,583	5,639
How to make a cake	94	14,923	2,056
Find a wedding videographer	98	19,425	2,532

3.3.2 The Web Datasets

The Web datasets were created by submitting queries to and collecting the returned Web documents from two search engines: Bing[1] and Google[2]. For each dataset, four to five queries related to the same topic were submitted to the search engines. For example, queries "trip to DC", "Washington DC", "DC", and "Washington" were submitted to create a dataset about the topic "plan a trip to DC". Each query contributes about 25 web documents; around 100 web documents are collected as a Web dataset.

In total, we created five Web datasets on the topics of *find a good kindergarten*, *purchase a used car*, *plan a trip to DC*, *how to make a cake*, and *find a wedding videographer*. For each dataset, we extracted about 40 concepts to organize them into concept hierarchies as testing tasks in the user study (Chapter 7). Table 3.2 presents the statistics of the Web datasets.

3.3.3 North American Industry Classification System (NAICS)

North American Industry Classification System (NAICS) is the industry standard used by the U.S. federal statistical agencies in classifying business establishments. In the latest 2007 version of NAICS, there are 92 top categories and 2,328 concepts in total. The relation described in NAICS datasets is *is-a*. We do not attempt to reconstruct the entire NAICS. We only extract several top categories of concepts from the 2007 NAICS as our datasets. Each category is one dataset. NAICS provides a well-developed ground truth to evaluate the quality of the constructed concept hierarchies. Unlike the public comments and the Web datasets, no document is available for this type of datasets. Instead, they only contain hierarchy fragments from the existing NAICS hierarchy.

[1]www.bing.com/.

[2]www.google.com/.

Table 3.3: Statistics of NAICS Datasets

Datasets/Concept hierarchies	**# concepts**	**Depth**	**The level w/ most concepts**
Information	40	4	3
Health care	40	3	3
Administrative services	40	4	3
Professional services	40	2	2
Finance	40	4	3
Construction	36	3	3
Public administration	35	2	2
Food manufacturing	40	3	3
Chemistry manufacturing	40	3	2
Merchant wholesalers	40	2	2

In total, we created ten NAICS datasets on the topics of *information, health care, administrative services, professional services, finance, construction, public administration, food manufacturing, chemistry manufacturing*, and *merchant wholesalers*. They are used as testing tasks for the user study. Each dataset contains about 40 concepts. Table 3.3 presents the statistics of the NAICS concept hierarchies.

3.3.4 WordNet

WordNet is a manually built English hierarchical dictionary. It consists of around 117,000 English words. We do not attempt to reconstruct the entire WordNet. Instead, we only extract some fragments from WordNet; each of these fragments is used as a dataset. They are used to evaluate the automatic concept hierarchy construction method, metric-based concept hierarchy construction (Chapter 5).

We extract both hypernym and meronym hierarchies from WordNet. The hypernym hierarchies indicate *is-a* relation among concepts; the meronym hierarchies indicate *part-of* relation among concepts. In particular, we only use one word sense within a particular hierarchy to avoid word sense ambiguity.

In total, there are 50 hypernym hierarchies and 50 meronym hierarchies extracted from WordNet. The 50 WordNet hypernym hierarchies are obtained from 12 topics: *gathering, professional, people, building, place, milk, meal, water, beverage, alcohol, dish,* and *herb.* The 50 WordNet meronym hierarchies are obtained from 15 topics: *bed, car, building, lamp,*

Table 3.4: 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

earth, television, body, drama, theater, water, airplane, piano, book, computer, and *watch.* Table 3.4 shows the hierarchy statistics, including the number of datasets, the total, average and maximum number of concepts, the average and maximum number of depth of the hierarchies, and the id of the level that containing that most number of concepts for the WordNet hierarchies.

3.3.5 ODP

Open Directory Project (ODP) is also a manually built concept hierarchy. It aims to organize the entire Web into hierarchies. Similar to the WordNet hierarchies, we only extract some fragments from ODP; each of these fragments is used as a dataset.

There is no meronym hierarchy available in ODP. Therefore we only extract hypernym hierarchies from ODP. In particular, 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).*

We extract 50 ODP hypernym hierarchies 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 statistics about these ODP hypernym hierarchies are also listed in Table 3.4.

3.3.6 Summary

In total, we use 171 datasets from 5 different sources to evaluate the techniques presented in this dissertation research. Some are document collections and others are hierarchy fragments. Table 3.5 gives a quick summary of these datasets.

Table 3.5: Summary of Datasets.

Datasets	Type	# datesets	Relation	Purpose	Domain	Chapter
Public comments	document	6	*user-defined*	tool training, user study	emails	4, 6, 7
Web	document	5	*user-defined*	user study	Web search	6, 7
NAICS	hierarchy	10	*is-a*	user study	industry standards	6, 7
WordNet	hierarchy	100	*is-a*, *part-of*	hierarchy reconstruction	general domain	5
ODP	hierarchy	50	*is-a*	hierarchy reconstruction	Web classification	5

3.4 Measuring Hierarchy Similarity

Measuring the quality of a concept hierarchy is challenging. To effectively evaluate our techniques, we review existing methods of hierarchy evaluation and present them in this section.

Concept hierarchies are considered as trees in this dissertation. We therefore first look at how to measure tree similarity. Specifically, concept hierarchies can be considered as *labeled* and *unordered* trees. “Labeled” means that every node in the hierarchy has a name. We often see “unlabeled” clusters generated by a hierarchical clustering algorithm and only the nodes at the leaf level have names. In a “labeled” tree, all nodes, including both leaf and non-leaf nodes, have names. “Unordered” means no ordering constraints are posed within the sibling nodes. This property makes it more challenging to discover the similarities or differences for the “unordered” trees than for the “ordered” trees. This is because there are $n!$ orderings among n sibling nodes in an unordered tree whereas there is only a single ordering among them in an ordered tree.

There are direct and indirect approaches being proposed to measure the similarity between concept hierarchies. The direct measurements include Tree Edit Distance and schema similarity measures. The indirect methods that are mainly used by the Information Retrieval (IR) research community to evaluate the browsing tasks.

3.4.1 Tree Edit Distance

Tree is a commonly used data structure in computer science and other disciplines such as biology. Research about tree similarity mainly focuses on identifying matches among the trees from the topology perspective. Tree Edit Distance is generally accepted as the standard method for measuring differences between trees. However, Tree Edit Distance is often infeasible to be used in real-life applications for the inefficiency issue. In fact, [ZJ94] has shown that computing Tree Edit Distances is not only NP-hard, but even a MAX SNP-hard problem.

Tree Edit Distance is defined on the editing operations. The basic tree edit operations include relabelling a node (Figure 3.2 (a)), inserting a new node (Figure 3.2 (b)), and deleting a node (its children become children of its parent node) (Figure 3.2 (c)). Recently, researchers also consider a move operation (move a subtree from one parent to a different parent) as the fourth edit operation for trees (Figure 3.2 (d)). In this dissertation research, we refer to the Tree Edit Distance which includes the relabelling, insertion, deletion, and move operations.

Formally, Tree Edit Distance between two trees T_1 and T_2 is defined as the minimum cost of turning T_1 into T_2 via a sequence of edit operation functions. The sequence of the edit operation functions is called an *edit script*. Tree Edit Distance is the minimum sum of the cost of every relabelling, deleting, insertion, and move operations for the mapping between the two trees.

Tree Edit Distance is usually applied to labeled trees, which can be further broke down into ordered and unordered labeled trees. In this dissertation research, we focus on the "unordered" labeled trees. A mapping between two unordered trees T_1 and T_2 happens when both the "one-to-one" condition and the "ancestor" condition are satisfied. The "one-to-one" condition says that an arbitrary node v in T_1 is found in T_2 as w. The "ancestor" condition says that an ancestor of v in T_1 has the same label as an ancestor of w in T_2. The third condition is the "sibling" condition, which constrains the order of siblings in a branch. However, it only applies to the "ordered" trees, not the "unordered" trees. This more restricted sibling condition makes the similarity calculation easier as compared to the "unordered" trees.

Most research on Tree Edit Distance was conducted for ordered trees for their relative simplicity. With the help of dynamic programming, ordered Tree Edit Distance can be solved

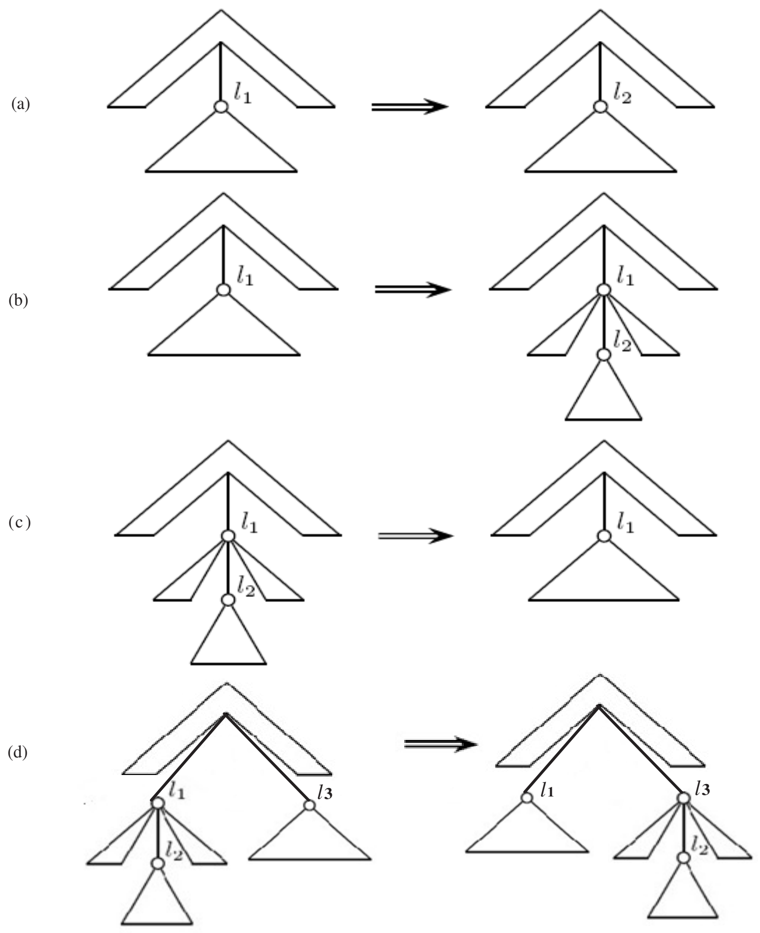


Figure 3.2: Tree Edit Operations. (a) Renaming node l_1 to l_2. (b) Inserting node l_2 as the child of l_1. (c) Deleting node l_2 from l_1. (d) Moving node l_2 from l_1 to l_3.

in polynomial time [Bil05]. The popular algorithms include Klein's algorithm [Bil05] and Zhang and Shasha's algorithm [ZS89].

For unordered trees, calculation of Tree Edit Distance is more complicated and much more inefficient. In fact, unordered Tree Edit Distance has been shown to be NP-complete [ZSS92] even for binary trees with only two labels. Furthermore, it has been shown to be MAX-SNP hard [ZJ94]. That means, unless $P = NP$, there is no polynomial-time approximation scheme (PTAS) for the problem. Only under special cases, such as T_2 only contains logarithmic number of leaves, there exists polynomial-time solution to calculate the unordered Tree Edit Distance for T_1 and T_2 [ZSS92].

Since calculation of Tree Edit Distance, especially unordered Tree Edit Distance, is computationally intractable, much research focuses on finding efficient approximations for Tree Edit Distance. Among them, many are for approximating the ordered Tree Edit Distance and a few for unordered Tree Edit Distance. We describe two approaches which are related to our method.

Yang et al. presented an approximation method of Tree Edit Distance for ordered trees [YKT05]. Their work transformed tree-structured data into binary branches by representing the occurrences of the binary branches as a numerical vector. L_1 norm of the vectors were calculated as the distance between two trees. This work characterized a tree as a set of q-level binary branches. Within a Tree Edit Distance of d, two trees share $[4(q-1)+1]d$ q-level binary branches. The authors proved that L_1 norm of the vectors is a lower bound of Tree Edit Distance. The computational complexity of the L_1 norm is $O(|T1| + |T2|)$. This approach is similar to our approach in terms of representing trees into vectors and using vector similarity measure as the tree similarity metric. However, the approach is different from ours in the method of mapping trees into vectors and the choice of vector similarity measure. Most importantly, instead of the ordered trees, we deal with the more challenging *unordered* Tree Edit Distance problem.

Kailing et al. presented an approach for unordered Tree Edit Distance in [KpKSS04]. They presented trees as histograms. Three kinds of histograms, including histogram of height of nodes, histogram of degree of nodes, and histogram of node labels, were employed. The authors used the maximal value among the L_1 norms calculated for the histograms as the distance between two trees. They proved that this maximal value is a lower bound of Tree Edit Distance between two unordered trees. The histogram of node labels is the piece of work that is the most similar to our approach. However, their work divided the range of node labels into bins (it isn't clear how they divide the node labels into bins). Our approach did not divide the terms into groups to best preserve the original form of the text. Instead, we employ common text preparation techniques, such as stop word removal and stemming, to create the vocabulary.

3.4.2 Schema Measurement

COMA++ is a tool for matching schemas and complex ontologies [ADMR05]. In [RhDM04], an earlier version of COMA++, the authors explained COMA++'s fragment-based schema matching approach. The similarity measure for hierarchy fragments in COMA++ is mainly based on metadata information including the schema types and metadata such as addresses and customers. However, such metadata information is usually not available in a personal concept hierarchy and hence not applicable to our task.

3.4.3 Indirect Measurement

The indirect measurements for hierarchy similarity are mainly used by the IR research community. When ontologies are used to represent and organize documents, how well a concept hierarchy is established for these documents can be indirectly evaluated by the effectiveness of finding related documents using this hierarchy. They are not intended to compare two concept hierarchies. Instead, they evaluate the quality for a very well-defined task - finding documents. Therefore, indirect measurements only apply when a task is known in advance.

Lawrie and Croft evaluated the effectiveness of a concept hierarchy by estimating the time it took to find all relevant documents [LC03]. Their work calculated the total number of menus that must be traversed and the number of documents that must be read as the metrics for hierarchy effectiveness.

Allan et al. studied how to evaluate hierarchical clusterings for the task of topic detection and tracking (TDT) [AFB03]. The authors proposed two metrics. The first was called "minimal cost", which performed a best-first search to locate a node that associates with the lowest cost. When determining the cost, this measure used a linear combination of C_{branch} (the number of child nodes) , C_{title} (the cost of reading a document title), and C_{det} (the number of edges travelled). The second measure was "expected travel cost", which accumulated the costs of reading titles in subtrees, parents, and sibling subtrees of a relevant document, and then averaged the cost over the number of on-topic stories. This measure claimed to be more user-oriented since it mimicked how a user searches in a hierarchy. However, it is also computationally expensive due to large branching factors and recursions of computations within concept hierarchies.

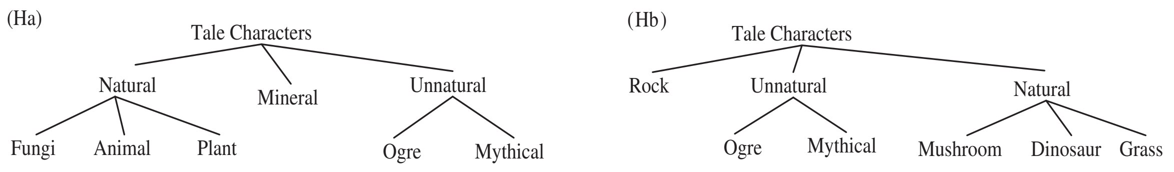


Figure 3.3: Hierarchy Examples ("tale characters").

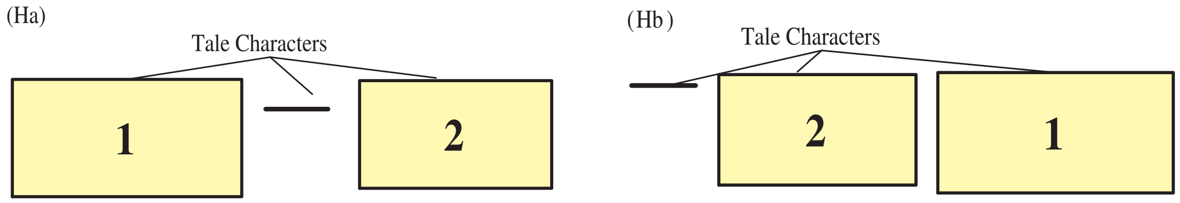


Figure 3.4: Fragment View of Concept hierarchies ("tale characters").

These indirect measures do not utilize the full hierarchical structures but only how documents are grouped at the leaf level. Moreover, they focus more on grouping and navigating to relevant documents, which is a different task from concept hierarchy construction. Therefore their evaluation approaches are not applicable to our problem.

3.5 Fragment-Based Similarity

We find that existing approaches are either too inefficient or not applicable to hierarchy evaluation. A good, practical hierarchy/hierarchy similarity measure is needed.We consider such a measure should meet the following desirable properties:

- The calculation of the metric should be efficient.
- It should be able to rank concept hierarchies based on their similarities as compared to a gold standard.

Concept hierarchies can be similar in terms of content, topology, and both. Content similarity refers to the similarity of node labels. Topology similarity refers to the similarity of the hierarchy structure and positions of nodes in the hierarchy. Both content and topology are important to determine how similar two concept hierarchies are.

Tree Edit Distance compares topology similarity and content similarity at a micro level. It counts every single insertion, deletion, move, and relabelling operation into the distance score for two concept hierarchies. To obtain the minimum distance of every possible edit scripts, it also needs to examine many combinations of a series of operations exhaustively. This is one of the reasons why Tree Edit Distance is computationally intractable.

On the contrary, people tend to compare hierarchies at a macro-level. We observe that when people compare two hierarchies, it is the *fragments of similar concepts* that first capture their eyes. For example, Figure 3.3 presents two ontologies, H_a and H_b. Comparing these two hierarchies, we can easily identify two distinct fragments that are rooted from "natural" and "unnatural" in both hierarchies. We then compare the nodes within each fragment to conclude whether the two hierarchies are similar. Moreover, the nodes outside the fragments, for example, "mineral", might be ignored or valued less in the comparison. Figure 3.4 illustrates the fragment view of Figure 3.3. Based on this observation, we are inspired to propose a new approach to compare hierarchies fragment by fragment. We call our approach *Fragment-Based Similarity (FBS).*

The idea is supported by the "subgraph isomorphism" approach in graph similarity. Subgraph isomorphism compares graphs *subgraph by subgraph* and considers "two graphs are similar if they are the same or one is contained in the other" [BDKW07]. This existing graph similarity measure supports our intuition that it is sensible to compare hierarchies fragment by fragment.

Based on these considerations, we take a fragment-by-fragment approach to calculate the similarity between hierarchies. In the following sections, we will use "fragment", "subtree", "branch", and sometimes "non-leaf node" interchangeably to refer to fragments in a concept hierarchy.

3.5.1 Vector Representation of Hierarchies

The bag-of-word model is widely used in IR for document retrieval, classification, and clustering. In this subsection, we elaborate how to use the bag-of-words representation to transform hierarchies into vectors and how this representation can help in a fragment-based similarity measure.

For a concept hierarchy T, a concept/a node label is treated as a term in the bag-of-word model, where a term can be a word, a phrase, or a concept. Particularly, a node is represented by a vector consisting of all the nodes in the subtree rooted from it. For a leaf

	tc	*natural*	*fungi*	*animal*	*plant*	*mineral*	*unnatural*	*rock*	*ogre*	*mythical*	*mushroom*	*dinosaur*	*grass*
tc	0	1	1	1	1	1	1	0	1	1	0	0	0
natural	0	0	1	1	1	0	0	0	0	0	0	0	0
unnatural	0	0	0	0	0	0	0	0	1	1	0	0	0

Figure 3.5: Vector representation for hierarchy H_a (tc means “tale characters”).

	tc	*natural*	*fungi*	*animal*	*plant*	*mineral*	*unnatural*	*rock*	*ogre*	*mythical*	*mushroom*	*dinosaur*	*grass*
tc	0	1	0	0	0	0	1	1	1	1	1	1	1
natural	0	0	0	0	0	0	0	0	0	0	1	1	1
unnatural	0	0	0	0	0	0	0	0	1	1	0	0	0

Figure 3.6: Vector representation for hierarchy H_b (tc means “tale characters”).

node, its subtree is itself. With the bag-of-word representation, the ordering among concepts within a subtree is ignored, hence the unordered nature of hierarchies are preserved.

Formally, suppose T_i and T_j's sizes are M and N respectively. Assume T_i consists of U non-leaf nodes and T_j consists of V non-leaf nodes. Using the vector representation, T_i is transformed into U vectors; each vector is of length $Z = |T_i \cup T_j| < M + N$. Similarly, T_j is transformed into V vectors and each vector is of length Z too.

We define the *vocabulary* Z of two hierarchies as the union of all the concepts in two hierarchies. For example, the vocabulary of H_a and H_b (Figure 3.3) , i.e. the union of their concepts, including both leaf and non-leaf nodes, is:

> {*tale characters, natural, fungi, animal, plant, mineral, unnatural, rock, ogre, mythical, mushroom, dinosaur, grass*}.

Figure 3.5 and Figure 3.6 demonstrate the vector representations for hierarchies H_a and H_b. A concept c_i in these two hierarchies is represented by a vector of 1s and 0s to indicate presence or absence of a concept in the subtree led by c_i, excluding c_i itself. For example, the vector representing the non-leaf node “natural” in hierarchy H_a displays 1s for “fungi”, “animal”, and “plant”, and 0s for the other terms in the vocabulary. The vector representing the non-leaf node “natural” in hierarchy H_b displays 1s for “mushroom”, “dinosaur”, and “grass”, and 0s for the other terms in the vocabulary. The length of the vectors equals to the size of the vocabulary.

We exclude a non-leaf node from its own vector representation. Although a non-leaf node trivially belongs to its own subtree, inclusion of itself in the similarity calculation may under-estimate the similarity between two fragments that only differ in the root (the non-leaf node). If not excluding the root and there are only a few nodes in a fragment, such as four nodes in the fragment "nature" in H_a, the different roots make a big impact (decrease) in the similarity calculation.

In summary, a concept hierarchy can be represented by a set of vectors. Each vector corresponds to a non-leaf node. The vectors consist of 1s and 0s indicating concept presence and absences in the subtrees. Once two hierarchies T_i, and T_j are represented as vectors, the calculation of FBS between them can be carried out in two steps: identifying matching fragments and aggregating the similarity scores.

3.5.2 Identifying Matching Fragments

To identify matching fragments, we first measure similarity between the fragments in T_i and the fragments in T_j by calculating the cosine similarities for these fragments. The similarity scores help us to align the fragments in the two hierarchies.

Calculating Subtree Similarity

Cosine similarity measures the angle between two vectors and is calculated by dividing the inner product of two vectors by the product of the vectors' lengths. Cosine similarity [MRS08] between two subtrees $t_p \subseteq T_i$, $t_q \subseteq T_j$ is calculated as follows:

$$sim_{cos}(t_p, t_q) = \frac{\overrightarrow{t_p} \bullet \overrightarrow{t_q}}{\|\overrightarrow{t_p}\|\|\overrightarrow{t_q}\|} = \frac{\sum_{k=1}^{Z} c_{pk} \times c_{qk}}{\sqrt{\sum_{k=1}^{Z} c_{pk}^2} \times \sqrt{\sum_{k=1}^{Z} c_{qk}^2}} \tag{3.3}$$

where c denotes a term/concept in a subtree, $Z = |T_i \cup T_j|$ is the vocabulary size, $\bullet$ denotes the vector dot product, and $\|\|$ is the length of a vector. The cosine similarity values range in [0,1], where a value of 0 means that two subtrees are unrelated and a value close to 1 means that the subtrees are closely related. It is obvious that in order to have cosine similarity greater than 0, two subtrees should have some concepts in common. When the concepts' presences are the same in two subtrees, the cosine similarity between these two subtrees is equal to 1. Figure 3.7 shows the cosine similarity values calculated for all the subtrees for H_a and H_b.

We compare every non-leaf node in T_i with every non-leaf node in T_j by cosine similarity.

	tale character(H_b)	natural(H_b)	unnatural(H_b)
tale character(H_a)	0.5	0	0.5
natural(H_a)	0	0	0
unnatural(H_a)	0.5	0	1

Figure 3.7: Cosine similarity between non-leaf nodes in H_a and H_b.

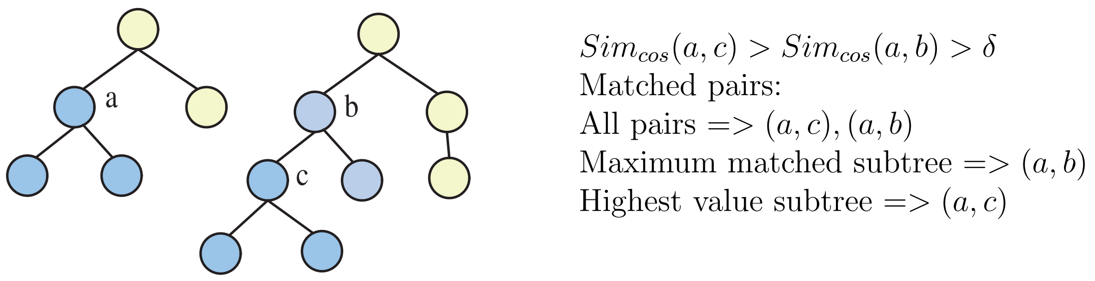
Thus there are UV cosine values computed in these two hierarchies. Looping over each pair of non-leaf nodes gives a complexity of $O(UV)$. Assume that there are Z concepts in the vocabulary, the calculation of cosine similarity between two term vectors gives a complexity of $O(Z)$. Hence the time complexity of this step is $O(UVZ)$. In the worst case, the number of non-leaf nodes is close to the hierarchy size, that is to say, $U = M$ and $V = N$. In this case, the time complexity can be written as $O(MNZ)$. If M, N, and Z are equal (since vocabulary is the union of two concept sets), the worse case time complexity is $O(N^3)$. In practice, however, since the vectors are sparse, the time complexity is usually much lower than $O(N^3)$.

Choices of Matching Criterion

After obtaining similarity values for every pair of fragments in T_i and T_j, we can align the fragments in these two hierarchies. Basically, for each fragment in T_i, we try to find its most similar fragment(s) in T_j. There are several options that can be used as the matching criterion:

All pairs. All pairs of fragments whose cosine similarity values are above a threshold δ are considered a match and hence are aligned. For those pairs whose cosine similarity values below δ, they should not be considered as a match because they are not similar to each other. The advantage of this option is its simplicity. The disadvantage is that there may exist many subtrees in T_i matched with a subtree in T_j. Multiple matches for a single subtree may produce ambiguous alignments.

Maximum matched subtree. With this option, every fragment/subtree in T_i looks for a maximum matched subtree in T_j. The *maximum matched subtree* for a subtree t_p in T_i is a subtree that is the largest subtree (i.e., a subtree that contains the most number of nodes) in T_j and the subtree's cosine similarity to t_p is greater than a threshold δ. The intuition is that when multiple matches occur, we should choose the largest subtree that is similar



$Sim_{cos}(a,c) > Sim_{cos}(a,b) > \delta$
Matched pairs:
All pairs => $(a,c),(a,b)$
Maximum matched subtree => (a,b)
Highest value subtree => (a,c)

Figure 3.8: Matched fragments based on different criteria.

enough to t_p to ensure that the matching is not partial. This results only one subtree in T_j matched with t_p. For example, the maximum matched subtree for "unnatural" in H_a is "tale characters" in H_b; the maximum matched subtree for "tale characters" in H_a is "tale characters" in H_b.

Highest value subtree. For a subtree t_p in T_i, the *highest value subtree* is the subtree with the highest cosine similarity to t_p among all subtrees in T_j. The similarity also needs to be greater than a threshold δ. Each non-leaf node in T_i has exactly one matched fragments in T_j. This option guarantees that the matched pairs are the most similar pairs based on their cosine similarity. To illustrate, the highest value subtree for "unnatural" in H_a is "unnatural" in H_b, and the highest value subtree for "tale characters" in H_a is "unnatural" in H_b. Other pairs' similarities are 0 and not considered as matched.

Because the vector representations are sparse in general, many cosine similarity values are in the lower range of [0,1]. Empirically, we set the threshold δ as 0.2 for the above options.

Figure 3.8 further illustrates how the above three matching criteria work by showing two hierarchies T_i and T_j. Assume that node a belongs to T_i, and nodes b and c belong to T_j. We look for a match for node a in T_j. Suppose the cosine similarity calculation shows that $sim_{cos}(a,c) > sim_{cos}(a,b) > \delta$. Different matching criteria output different subtrees in T_j as the match for a. "All pairs" outputs two matched pairs for node a - (a,b) and (a,c) - since both b and c pass the selection threshold δ. "Maximum matched subtree" outputs b as the match for a since b is the largest subtree among all the subtrees passing δ (b and c). "Highest value subtree" outputs c as the match since $sim_{cos}(a,c)$ is the highest similarity value among that for all the subtrees passing δ. We evaluate and compare the above three options of aligning fragments empirically in Section 3.5.4.

3.5.3 Aggregating Similarity Scores

After calculating cosine similarities and identifying the matched fragments, we aggregate the values to produce a single similarity score for the two hierarchies as their FBS score.

Suppose there are m matched pairs of fragments from two hierarchies T_i and T_j, and the cosine similarity values for the matched fragment pairs are known. We aggregate the similarity scores by averaging them to obtain the final FBS score. The averaging can be done by dividing the sum of all cosine similarity scores of the matched pairs over a normalizing denominator D. Thus the overall FBS similarity of T_i and T_j can be written as:

$$FBS(T_i, T_j) = \frac{1}{D} \sum_{p=1}^{m} sim_{cos}(t_{ip}, t_{jp}) \tag{3.4}$$

where $t_{ip} \subseteq T_i$, $t_{jp} \subseteq T_j$, and they are the p^{th} matched pair among the m matched fragment pairs, D is the denominator.

Therefore, the aggregated FBS score consists two parts: the sum of scores of each matched pairs and D. Since the matching is done, the sum of scores of matched pairs are known. However, D could be computed in several different ways.

Choices of Denominator D

A natural choice of the denominator D is the number of matched pairs in T_i and T_j. Thus Equation 3.4 becomes:

$$FBS(T_i, T_j) = \frac{1}{m} \sum_{p=1}^{m} sim_{cos}(t_{ip}, t_{jp}) \tag{3.5}$$

where m is the number of matched pairs in T_i and T_j. However, averaging by m may produce false positives when a few small fragments matched with high cosine similarity values. For example, suppose T_i is about "education" and T_j is about "war". T_i's size is $M = 200$, and T_j's size is $N = 100$. T_i and T_j are not similar to each other because they are very different in topics and vocabulary. However, there exist two matched pairs of fragments between them. The two matches are in small-size ($<< M, << N$), but with high cosine similarity matching values - 0.9 and 0.8. In this case, $m = 2$, the sum of the similarity values is $0.9 + 0.8 = 1.7$, and the FBS according to Equation 3.5 is 0.85. Such a high FBS score suggests that the similarity between T_i and T_j is high, however, because the matched fragments only account for a small portion of the entire hierarchy, the true similarity should be low. In this case, using the number of matched pairs as the denominator generates false positives.

Another choice for D is the hierarchy size. Suppose T_i's size is M, and T_j's size is N. When $M = N$, it is straightforward for D to be M. However, when $M \neq N$, we may take either the *min* or *max* of M and N as D. One may assume that a small hierarchy and a big hierarchy's similarity is 1 if the small one is a subgraph within the big one. Based on that, *min* of M and N could be a good denominator. However, in this dissertation, we assume that two hierarchies are not similar if they are very different in sizes because the chance that they contain different concepts is high. Based on this assumption, we choose the *max* of M and N as the denominator D. Thus, Equation 3.4 becomes:

$$FBS(T_i, T_j) = \frac{1}{max(M, N)} \sum_{p=1}^{m} sim_{cos}(t_{ip}, t_{jp}) \tag{3.6}$$

where M is the number of nodes in T_i, N is the number of nodes in T_j, and m is the number of matched pairs of fragments.

As every subtree corresponds to a non-leaf node, the third choice of D is the number of non-leaf nodes in a concept hierarchy. Suppose T_i contains U non-leaf nodes, and T_j contains V non-leaf nodes. For a similar reason as mentioned for the second option, we take the maximum value of U and V as the denominator. Thus Equation 3.4 becomes:

$$FBS(T_i, T_j) = \frac{1}{max(U, V)} \sum_{p=1}^{m} sim_{cos}(t_{ip}, t_{jp}) \tag{3.7}$$

where U is the number of non-leaf nodes in T_i, V is the number of non-leaf nodes in T_j, and m is the number of matched pairs. In Section 3.5.4, we evaluate the above options of the denominator D for aggregating the similarity scores between two hierarchies.

Note that cosine similarities between the matched fragment pairs have been calculated in the previous step. Thus the aggregation only needs to calculate an average and its time complexity is $O(1)$. Therefore the overall time complexity of calculating FBS for two hierarchies remains as $O(N^3)$. If pairwise node similarities are pre-calculated, the time complexity can be further reduced to $O(n)$.

We use FBS to directly evaluate the quality of the hierarchies built automatically against gold standards in Chapter 5, as well as use it to find out how similar and how different people construct hierarchies and whether they are self-consistent in Chapter 6 and Chapter 7.

3.5.4 Experiments

In this section, we empirically evaluate FBS. We show that it is a good approximation of Tree Edit Distance for unordered trees. Moreover, it is much more efficient than Tree Edit Distance and can be used for large hierarchies. We first describe a simulation process of Tree Edit Distance and based on that we create an evaluation dataset. We then examine the options of FBS and show that FBS is able to approximate Tree Edit Distance by generating highly correlated similarity rankings.

Datasets and Experimental Setup

It is a NP-complete problem to compute Tree Edit Distance between two trees. It is often infeasible to apply it in real applications on real data. Due to its inefficiency, instead of actually computing Tree Edit Distance, we simulate it by tracking the edit operations that are automated by computer. We create edit scripts by sequentially editing a tree and recording the edit operations during the process. Tree Edit Distance simply equals to the length of the edit script.

The simulation employs two functions, namely RandomTreeCreation and RandomTreeEdit. The RandomTreeCreation function creates random unordered labeled trees of a given tree size n. It takes in two inputs, n and a maximum branching factor b_{max}, and outputs a random labeled tree with n nodes. Each non-leaf node in this tree randomly branches up to b_{max} children. We call the hierarchies generated by RandomTreeCreation as "reference hierarchies". b_{max} is set to 5 in this process.

The RandomTreeEdit function makes random edits to an existing tree. It takes in two parameters - the total number of edit operations N_{edit}, and an existing tree T. It outputs a tree T' modified from T with N_{edit} edit operations. We assume that N_{edit} is the minimum number of edit operations turning a reference hierarchy into a modified hierarchy. Obviously, N_{edit} equals to the Tree Edit Distance. This function performs four basic edit operations, including insertion, deletion, relabelling, and move. Based on a user study which involved 24 users (Chapter 7), the ratio of the editing operations insertion:deletion:relabeling:move by real users is about 1:1:1:9. Within the N_{edit} edit operations, the percentage of different edit operations preserves this ratio.

We use RandomTreeCreation to create the reference hierarchies with 980 different sizes. The sizes range from 20 to 1000. For each size, we create 50 random reference hierarchies. We then use RandomTreeEdit to generate 10 modified hierarchies for each reference hierarchy. Suppose a concept hierarchy's size is M, the 10 modified hierarchies are created with N_{edit}=1, *.1M, .2M, .3M,.4M, .5M, .6M, .7M, .8M*, and *.9M*, respectively.

In the experiments, we illustrate that FBS is a good approximation of Tree Edit Distance. We compute similarity between many pairs of hierarchies using both Tree Edit Distance and FBS. We then rank these hierarchies based on FBS and Tree Edit Distance and show that the ranked lists generated by these two methods are highly correlated. We then study and compare the different options of the matching criterion, denominator D, and conclude the best options for FBS.

Correlation of FBS and Tree Edit Distance

FBS and Tree Edit Distance cannot be directly compared because they have different value ranges. FBS ranges from 0 to 1, while Tree Edit Distance ranges from 0 to ∞. Moreover, FBS measures similarity while Tree Edit Distance measures dissimilarity. A bigger FBS value means that two hierarchies are more similar, but a bigger Tree Edit Distance value means that they are less similar. Due to the above reasons, we compare the two methods via comparing the similarity rankings that generated by them, instead of a direct comparison of the values.

As described earlier, Tree Edit Distance scores are generated by simulations. FBS scores are generated based on the algorithm presented in this section. We calculate FBS for each pair of reference hierarchy and a modified version of it.

To evaluate whether the ranked lists generated by FBS and Tree Edit Distance are correlated, we employ Spearman's rank correlation coefficient ρ [WMS02] to compare the resulting ranked lists. Generally speaking, the higher the correlation between the rankings by a method and by the "gold standard", the better the method is concluded to be. In

Tree Size	20	50	100	500
Denominator = # matched pairs				
Match=All Pair	0.75	0.83	0.87	0.89
Match=Maximum Subtree	0.79	**0.90**	0.91	0.93
Match=High Value	**0.80**	**0.90**	**0.92**	**0.94**
Matching Criteria =highest value				
Denominator=#Matched	**0.80**	0.88	0.89	0.94
Denominator=#Non-leaf	**0.80**	0.89	0.89	0.90
Denominator=Hierarchy Size	**0.80**	**0.90**	**0.92**	**0.97**

Table 3.6: Spearman's ρ of FBS and Tree Edit Distance (Bold font indicates the best performance in a column).

our case, the "gold standard" is Tree Edit Distance. Pairs of rankings whose Spearman's ρ values are at or above 0.9 are often considered as "effectively equivalent" [Voo01]. Given N raw values X_i, Y_i, their ranks x_i, y_i, and the rank differences $d_i = x_i - y_i$, the Spearman's ρ value (no tie) for two ranked lists is:

$$\rho = 1 - \frac{6 \times d_i^2}{N(N^2 - 1)} \tag{3.8}$$

Table 3.6 displays Spearman's ρ values for the ranked lists generated by FBS and by Tree Edit Distance averaged over 50 random hierarchies at different sizes. We pick hierarchies in size of 20, 50, 100, and 500 as the representatives among the 980×50 randomly reference hierarchies. We choose to these sizes because they are close to the sizes of the hierarchies used in the user study. For hierarchies with other sizes, we observe the same trend. Bold font indicates the best performance in a column.

Since there are different options of FBS, we first calculate FBS based on different matching criterion while fixing the denominator to be the number of matched pairs. We then calculate FBS using different denominators while fixing the matching criterion to be "highest value subtree". In most runs, the correlations between FBS and Tree Edit Distance are very high (> 80%). The results show that the best matching criterion is the "highest value subtree" and the best denominator is "hierarchy size". The best runs (bold fonts in Table 3.6) achieve an averaged ρ=0.97, which indicates that FBS is a good very approximation to Tree Edit Distance for rank similarities.

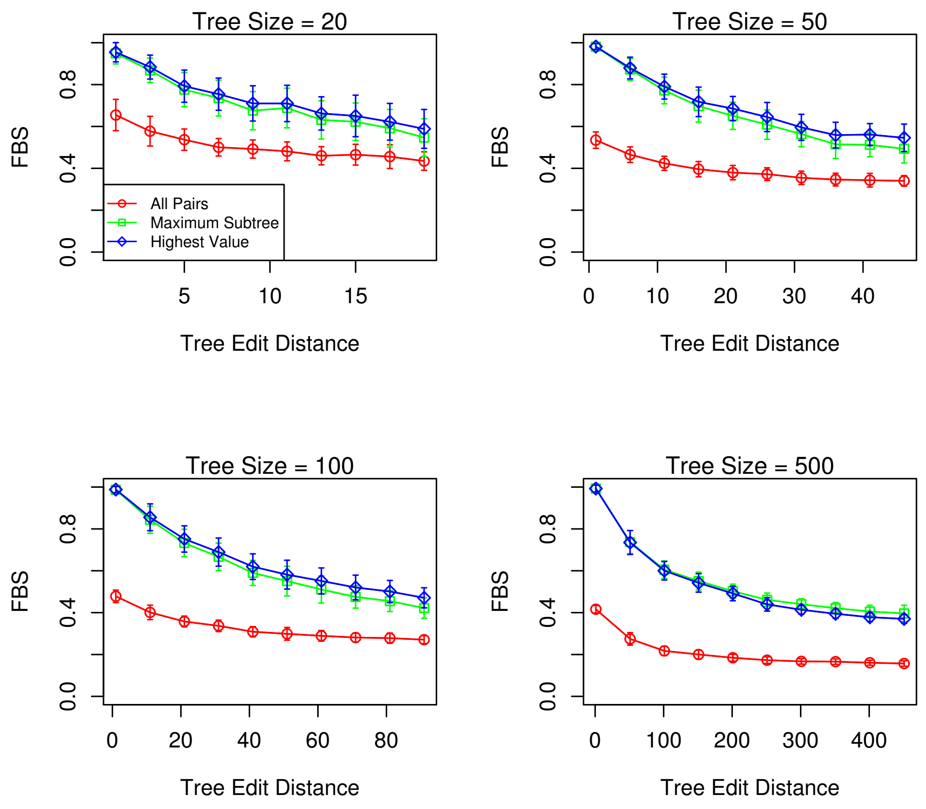


Figure 3.9: Fragment matching using different matching criteria (Denominator = #matched pairs).

Impact of the Matching Criteria

To further illustrate how well FBS approximates Tree Edit Distance as well as FBS's behavior under different matching criteria, we plot FBS scores vs. Tree Edit Distances.

Figure 3.9 plots the mean values of FBS for 50 random runs for hierarchies with size 20, 50, 100, and 500. The denominator is fixed to be the number of matched pairs, m. The x-axis is the Tree Edit Distance scores, and the y-axis is the FBS scores. The plots show the corresponding FBS score for a Tree Edit Distance score. The Tree Edit Distance scores increase continuously. If FBS is a good approximation to Tree Edit Distance, the plots of FBS should be smooth. That is, it should be either monotonically increasing or monotonically decreasing. This is because if there are many disagreements between FBS and Tree Edit Distance, the plots will contain many jumps and not be monotonic or smooth. In another word, whether the FBS scores are high or low is less important as whether the plots are smooth.

Figure 3.9 plots the curves for the three matching criterion options: "all pairs", "maximum matched subtree" and "highest value subtree". The plots indicate that all options are able to approximate Tree Edit Distance well by showing smooth curves.

We also observe that matching fragments based on "maximum matched subtree" and "highest value subtree" produce similar FBS values. While matching fragments based on "all pairs" produce FBS in a lower value range. Because in "all pairs", for a fragment there exist multiple matches and among which there are repeated matches with low cosine similarity values. The plots suggest that both "maximum matched subtree" and "highest value subtree" are good matching criteria to select a matched fragment pair. The latter option shows a high correlation score in Table 3.6, therefore we recommend it as the matching criterion for fragment alignment.

The error bars in Figure 3.9 indicate the standard deviations. The tighter the error bars, the more confident the approximation. When hierarchy size is big (tree size=500), the approximation is even better with smaller variance. When hierarchy size is small (tree size=20), the curve is less smooth and the approximation is less reliable. Although Figure 3.9 only shows hierarchies in four different sizes, we observe the same trend in hierarchies with other sizes.

Impact of the Denominator

After deciding the best matching criterion, we next determine the best denominator for FBS calculation. We have presented three options for the denominator D: "the number of matched pairs", "the number of non-leaf nodes", and "the hierarchy size". In Figure 3.10, we plot the corresponding FBS scores for continuously increasing Tree Edit Distance scores and observe whether the plots are smooth.

Figure 3.10 shows that all options for the denominators enable FBS to well-approximate Tree Edit Distance because the plots are all smooth. Moreover, we find that when using "hierarchy size" as the denominator, FBS results at a range of lower values. It is because that a hierarchy's size is usually much bigger than the number of matched fragments and

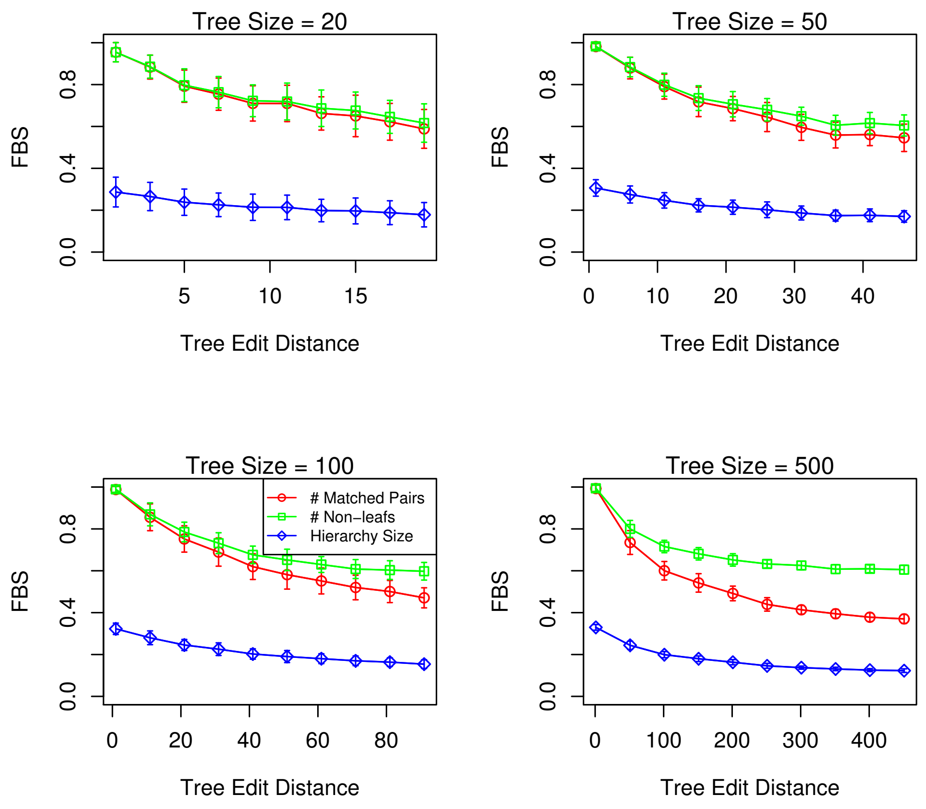


Figure 3.10: Similarity aggregation using different denominators (Matching=highest value).

the number of non-leaf nodes within it. Thus, as a bigger denominator, the hierarchy size produces low absolute values. However, as we mentioned before, it does not matter whether the curve is high or low and the values are big or small. What matters is whether the plots are smooth.

In addition, the error bars in Figure 3.10 are tight for the curves. It again indicates that FBS is a good approximation to Tree Edit Distance. As the hierarchies' sizes become bigger, the error bars become tighter, which indicates that the approximation works even better for bigger hierarchies.

Although Figure 3.10 shows smooth curves from all three options, Table 3.6 indicates that "hierarchy size" achieves a higher correlation score and is a better choice for the denominator. Thus we choose the "highest value subtree" as the matching criterion and the "hierarchy size" as the denominator to calculate FBS.

3.6 Summary

This chapter presents the background knowledge for this dissertation research. It describes the problem definition, the hierarchy construction tool, the datasets and the evaluation metrics used in the dissertation research.

Specially, concept hierarchy construction can be defined as creating a concept hierarchy with personal preference. Starting from a set of documents, concept hierarchy construction help a user to sort through the materials in the document set, extract concepts from it, and organize the concepts based on their relations identified from the documents. The user's personal idea of how to organize these concepts is the guidance for a concept hierarchy construction system.

This chapter also describes OntoCop, an interactive tool to create hierarchies. It supports both manual and interactive constructions of hierarchies. This chapter presents its editing functions and its user interface. The more advanced interaction functions and how they work will be presented in Chapter 6.

Moreover, this chapter describes the 171 datasets used in this dissertation research. The "document collection" datasets contain no inherent concept sets and hence the concepts must be extracted from the documents. This type of datasets includes the emails sent from the public and well-edited Web pages as search results. The "reference hierarchy" datasets have well-defined concept sets and relations that represent a community agreement about how to organize information; they provide gold standards, but no documents. They are used to test to a community-agreed gold standard how close a constructed hierarchical organization can be. Using a variety of datasets enables us to better evaluate our system under various situations.

In addition, this chapter presents Fragment-Based Similarity (FBS), a simple and efficient solution to measuring hierarchy similarity. Instead of comparing the entire hierarchy, our approach compares fragments between hierarchies and aggregates their similarity values as the final score. A comparison between the new metric and Tree Edit Distance shows that the proposed metric can generate consistent rankings of similarities as Tree Edit Distance, but with a much efficient time complexity of $O(n^3)$ ($O(n)$ if pairwise node similarities are pre-calculated).

With all the background knowledge, we are now ready to explore the techniques and algorithms for concept hierarchy construction in the following chapters.

Chapter 4

Concept Extraction

A concept hierarchy consists of *concepts* and *relations* among the concepts. To construct a concept hierarchy, the first step is to determine the concepts. *Concept extraction* is the process of identifying concepts from an arbitrary document collection.

In concept hierarchy construction, for a given document collection (a domain), concepts refer to the topics of interest for a user who constructs the hierarchy. Usually, concepts are nouns or noun phrases since they are the topics, either subjects or objects, that people concern about. For instance, in the sentence "I strongly urge you to cut mercury emissions from power plants by 90 percent by 2008", the phrases that draw one's attention are probably "mercury emissions", "power plants", and "90 percent". Though we agree that verbs and adjectives also play important roles in conveying meanings, they don't determine the topic for a sentence and hence are not considered as concepts in this dissertation research.

Various approaches have been proposed to identify concepts from documents. Most research approaches adopt part-of-speech (POS) tagging. Other commonly used techniques include verb predicate extraction [Sab04, WVH06], term frequency [RM04], and topic centroid extraction [FMG05]. Recent studies, such as Never Ending Language Learner (NELL) [CBK$^+$10] and TextRunner [BE08], extract concepts by using open domain surface text patterns, however, they are not yet common.

In this chapter, we employ a simple but effective approach to identify concepts in three steps: *concept mining*, *concept filtering*, and *concept unification*. Our strategy is to first mine candidate concepts exhaustively and then select the good ones. We evaluate it by comparing the concepts with manually constructed gold standards and show that his approach achieves a reasonable accuracy.

4.1 Concept Mining

Concept mining aims to extract *all* possible concept candidates from a document collection. At this stage, we hope to extract as many concept candidates as possible while preserving a certain degree of accuracy. As described earlier, concepts are nouns or noun phrases. We therefore extract nominal unigrams, bigrams, trigrams, and even longer noun sequences as concept candidates.

To obtain nominal unigrams, bigrams, and trigrams, documents are split into sentences. Each sentence is labeled by a part-of-speech (POS) tagger[1], which 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 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 ./.

The tagged sentence is then scanned through a noun sequence generator to identify noun sequences, i.e., sequences 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.

From these noun sequences, we generate four types of n-grams, including unigrams, bigrams, trigrams, and longer n-grams (length>3). For each type of n-grams, the frequency of occurrences are counted and they are sorted into a ranked list in the descending order based on the frequency. Since more frequent n-grams are more likely to correspond to concepts, we can easily apply a cut-off threshold or select the top k n-grams from the ranked lists as initial concept candidates. Table 4.1 shows the top ranked bigrams and trigrams generated from the Mercury and the Polar Bear public comment datasets. In practice, we apply a frequency cut-off threshold of 5 to the ranked lists to select concept candidates.

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

Stats/Datasets	Mercury Dataset	Polar Bear Dataset
#unique bigrams	1,462	1,389
total # of bigrams	4,510	3,397
#unique trigrams	129	361
total # of trigrams	900	731
	power plants(370)	greenhouse gas(248)
	mercury pollution(250)	gas pollution(227)
Top 5 bigrams	mercury emissions(175)	sea ice(143)
	mercury levels(110)	ice habitat(117)
	clear air(70)	endangered species(115)
	clean air act(65)	greenhouse gas pollution(227)
	environmental protection agency(35)	sea ice habitat(115)
Top 5 trigrams	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 for Mercury and Polar Bear datasets. (The numbers in brackets indicate frequency.)

Frequent nominal n-grams are good concept candidates, however, sometimes the longer but infrequent n-grams can also be good concept candidates, such as the 7-gram noun sequence "*Toxics/NNP Release/NNP Inventory/NNP Burden/NNP Reduction/NNP Proposed/NNP Rule/NNP*". Such longer n-grams often capitalize their first letters in every word. We hence develop an efficient named entity (NE) recognizer for selecting concept candidates from longer n-grams (length>3). Regardless the frequency of a noun sequence, the NE tagger labels a noun sequence as a named entity if it capitalizes all the first letters. These named entities are considered as concept candidates as well.

The frequent nominal unigrams, nominal bigrams, nominal trigrams, and longer n-grams with capitalizations are selected as the initial concept candidates.

4.2 Concept Filtering

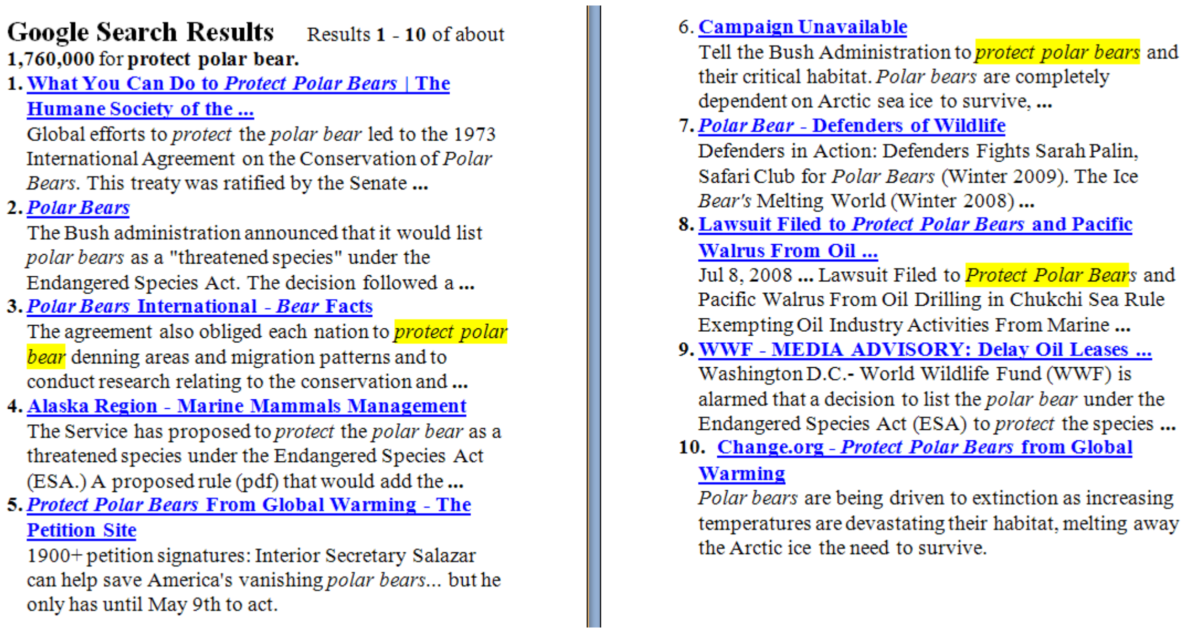
Concept mining produces many concept candidates. There are false positives among them. Many errors are caused by the POS tagger's mistakes in assigning labels. For example, *protect polar bear* is incorrectly labeled by the POS tagger as three nouns, and hence concept mining incorrectly returns *protect polar bear* as a candidate, however the correct concept should be *polar bear*. Moreover, when a document contains spelling errors, grammar errors, inappropriate capitalizations, or wrong punctuations, a POS tagger is more likely to make mistakes. To remove such errors in concept candidates, we perform *concept filtering*.

We observe that many false positive concept candidates contain a word which is not a noun. These concept candidates have 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. Recall that the desirable concepts are noun phrases, such as *polar bear* and *mercury pollution*. We just need to remove the verb or adjective.

We also observe that noun phrases are often used frequently in our daily language. Conversely, the joint occurrence of verb/adjective and noun phrase is used much less frequently. Therefore, we design a web-based frequency test to filter out the false positives. Particularly, we employ Google search results to evaluate and filter the concept candidates. Queries formed by each concept candidate are sent to the Google search engine. For example, the concept candidate "polar bear" is used as a search query on Google. Among the first 10 returned Google snippets (not including the titles nor the web page addresses), if a concept candidate appears more than a cut-off threshold number of times (the threshold is empirically set to 4), the concept candidate is considered as a commonly-used noun phrase, and hence a good concept; otherwise, an error.

Figure 4.1 shows the snapshot of the first 10 Google snippets for *protect polar bear*. Since the occurrences of this phrase is only 3 (< 4), the 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 snippets for another concept candidate *polar bear*. Since the occurrences of this phrase is 14 (> 4), the web-based frequency test judges it as a commonly-used noun phrase, thus a valid concept candidate, and is kept.

Through concept filtering, false positives caused by POS tagging errors are effectively identified and removed. Moreover, spelling errors, such as *polor bear*, *pulution*, are also removed. There are still other sources of errors in identifying concepts. For instance, personal preference could be a source of error. Particularly, when a concept could be valid but it is not interesting enough to be included in a hierarchy constructed by a user. It could be evaluated



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* (an error): 3 occurrences.

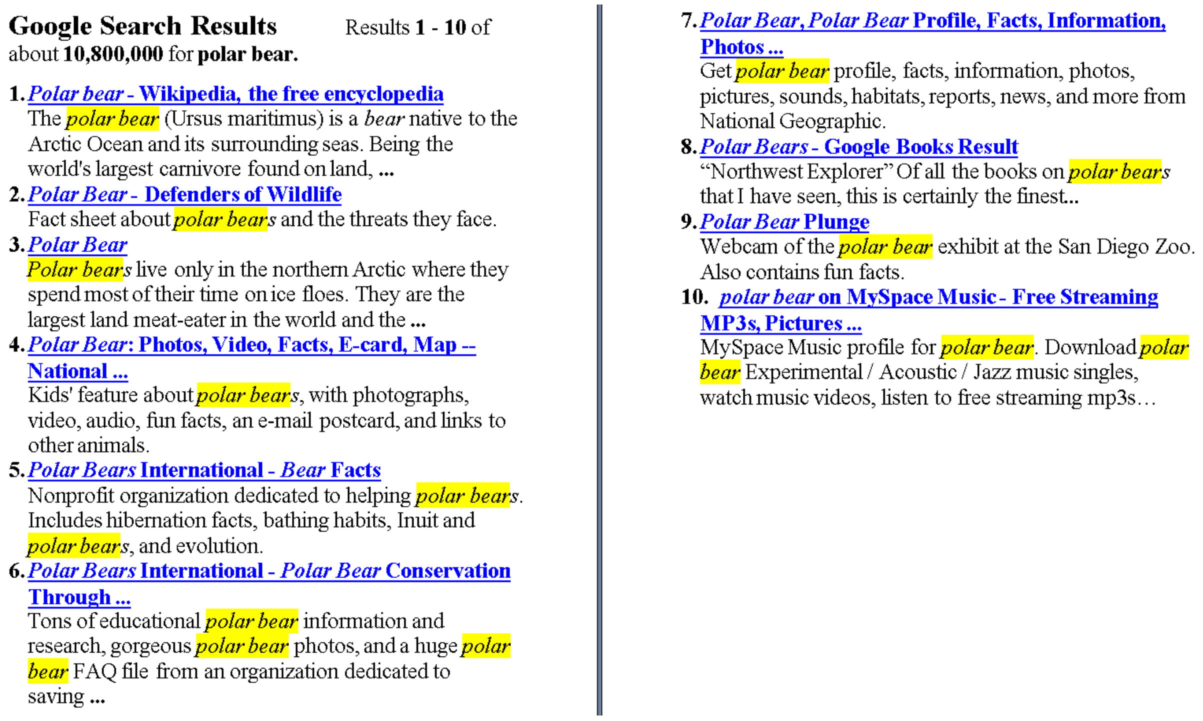
as invalid by this user.

4.3 Concept Unification

Another problem in concept extraction is that many concept candidates share similar meanings. For example, the top 2 and top 3 bigrams for the Mercury dataset, "mercury pollution" and "mercury emissions", actually refer to the same concept and only one of them needs to be kept. We address this issue by *concept unification*.

We employ Latent Semantic Analysis (LSA) [DDF+90] for concept unification. LSA is an Information Retrieval (IR) technique which is able to cope with two classic problems - synonym and polysemy. Technically, LSA maps a document collection's concept-document matrix from a large space to a lower-dimension space through singular vector decomposition (SVD) and dimension reduction. We recognize similar concepts from the current set of concept candidates based on the following steps:

1. Create a concept-document matrix C. Each entry C_{ij} in C represents the tf.idf weight [MRS08] of concept c_i in document d_j.


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 bears and the threats they face.
3. **Polar Bear**
Polar bears 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 bears, 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 bears. Includes hibernation facts, bathing habits, Inuit and polar bears, 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 bears 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* (a concept): 14 occurrences.

2. Perform SVD for C and obtain the SVD term matrix U, singular value matrix Σ, and SVD document matrix V. The rank of C is r, which indicates the first r non-zero diagonal entries in Σ.

3. Reduce the rank of C from r to k, where k is an integer much smaller than r. We empirically test different k values and choose the k value based on [TWH00]. Basically, k is selected if it generates an “elbow value” in the summary purity scores for the series of concept clustering results. k is tested from 50 to 500 in our experiments.

4. Given k, set the last $(r - k)$ diagonal entries in Σ to be zeros to obtain Σ_k. Calculate a truncated concept-document matrix at rank-k: $C_k = U\Sigma_k V^T$.

5. Compute the concept-concept matrix $C_k C_k^T$, whose $(m, n)^{th}$ entries indicates the similarity between two concepts c_m and c_n in the lower dimension space. Cluster the concepts based on the scores in $C_k C_k^T$. Repeat steps 3, 4 and 5 to pick the best k.

6. From each concept clusters, choose a concept with the highest corpus frequency to be

kept in the final concept set.

Note that the computation cost of SVD is high when the number of documents is large. Due to computational limitation, we build the LSA representation on a random sampled subset (500 to 800 documents) of a document collection.

4.4 Experimental Results

To evaluate the performance of concept extraction, we invited experts to create ground truth ontologies. 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 experiments. Twelve experts were all from Political Science background and were familiar with the problem domain participated in the experiment. They were asked to construct ontologies using OntoCop with the concept candidates produced by concept mining in a bottom-up fashion until they felt satisfied with their work or reached a 90-minute limit.

The experiments were conducted in the domain of public comments. The situation given in the evaluation was that the experts organized the comments into rule-specific ontologies based on their short-term needs. For instance, an expert wanted to know important concerns from the public about a rule proposed to protect polar bears. We use four public comment datasets in the experiments: Mercury, Polar Bear, Wolf, and TRI.

The quality of concepts are measured by comparing the concept candidates produced at each step in concept extraction and the ground truth. The evaluation metrics are Precision, Recall and F1-measures. Precision (P) is calculated as the number of correctly returned concepts divided by the number of returned concepts. Recall (R) is calculated as the number of correctly returned concepts divided by the total number of correct concepts in the ground truth. F1-measure is calculated as 2*P*R/(P+R).

In Tables 4.2, 4.3, and 4.4, we report the Precision, Recall and F1-measure after concept mining, concept filtering, and concept unification. The tables show how the Precision increases after each step in concept extraction.

Table 4.2 summarizes the performance of concept mining at different cut-off thresholds (2, 3, or 5) of frequency. Bold font indicates the best performance in a column. As the cut-off threshold increases, fewer concepts are kept, Precision increases, Recall slightly drops,

Table 4.2: Performance of Concept Mining.

Dataset	**#Concepts**	**P(Freq.≥2)**	**R(Freq.≥2)**	**F1(Freq.≥2)**
TRI	909	0.03	**0.55**	0.05
Wolf	6,446	0.03	**0.75**	0.06
Polar Bear	7,980	0.02	**0.70**	0.04
Mercury	23,605	0.02	**0.59**	0.03
	#Concepts	**P(Freq.≥3)**	**R(Freq.≥3)**	**F1(Freq.≥3)**
TRI	475	0.33	0.54	0.42
Wolf	3,302	0.43	0.71	0.54
Polar Bear	2,862	0.42	0.68	0.52
Mercury	11,126	0.38	0.57	0.46
	#Concepts	**P(Freq.≥5)**	**R(Freq.≥5)**	**F1(Freq.≥5)**
TRI	268	**0.53**	0.52	**0.51**
Wolf	1,645	**0.63**	0.71	**0.67**
Polar Bear	1,064	**0.62**	0.67	**0.64**
Mercury	4,148	**0.68**	0.56	**0.61**

Table 4.3: Performance After Concept Filtering.

Dataset	**#Concepts**	**P**	**R**	**F1**
TRI	128	0.76	0.52	0.62
Wolf	674	0.86	0.71	0.78
Polar Bear	568	0.80	0.67	0.73
Mercury	1,575	0.83	0.56	0.69

and F1 increases. Note that when the cut-off threshold changes from 2 to 3, Precision and F1 increase significantly. This result suggests that the concept candidates that appear at low frequency are mostly noise. When the cut-off threshold is set to 5, Precision of concept mining ranges from 0.53 to 0.68, Recall ranges from 0.52 to 0.71, and F1 ranges from 0.51 to 0.67. These results are moderately good, but not good enough to build a reasonable concept hierarchy.

We further evaluate the subsequent steps of concept extraction. We compare the concept candidates after performing concept filtering with the concepts produced by the experts. Table 4.3 shows the Precision, Recall and F1 after concept filtering. Particularly, Precision ranges from 0.76 to 0.83, Recall remains from 0.52 to 0.71, and F1-measures range from 0.62 to 0.78. The Precision shows a 22%-43% improvement and the F1-measure show a

Table 4.4: Performance After Concept Unification.

Dataset	**#Concepts**	**P**	**R**	**F1**
TRI	84	0.80	0.50	0.62
Wolf	117	0.88	0.68	0.77
Polar Bear	105	0.86	0.65	0.74
Mercury	423	0.85	0.56	0.68

13%-21% improvement over the best setting (frequency ≥ 5) in concept mining. The results demonstrate that concept filtering is effective. By removing tagging errors, concept filtering improves the overall concept extraction performance, especially Precision and F1.

We further compare the concept candidates after concept unification with the ground truth. Table 4.4 shows the Precision, Recall and F1 after concept unification. Precision ranges from 0.80 to 0.85, Recall drops a little and ranges from 0.50 to 0.68, F1-measures remain from 0.62 to 0.77. The Precision shows a 3%-8% improvement over concept filtering. These results demonstrate that concept unification effectively groups the similar concepts into single entries and further improves Precision of concept extraction.

4.5 Summary

This chapter describes our techniques for concept extraction. We present a simple but effective approach. It is performed in three steps: concept mining, concept filtering, and concept unification. Concept mining examines a given document collection and outputs a large set of concept candidates, including nominal unigrams, bigrams, and trigrams, and longer n-grams with capitalizations. Concept filtering evaluates the candidates' validity by conducting a web-based frequency test. It effectively removes invalid concepts created due to POS tagging errors. Concept unification further conflates similar concepts using LSA and outputs the unified concepts as the set of concept candidates.

By comparing the concepts obtained by concept extraction and the ground truth, we find that the remaining errors are basically concepts created by the human experts. Those concepts are related to the domain, however do not directly appear in the corpora.

Nonetheless, the techniques presented in this chapter are sufficiently accurate (Precision around 0.85 and Recall around 0.55 on average) for this dissertation research, which focuses more on organizing concepts into ontologies through relation formation. The following chapters describe the techniques for *relation formation*.

Chapter 5

Metric-based Concept Hierarchy Construction

Before the user participates in concept hierarchy construction and interacts with the system to give her preferences, it is desirable that the system can propose an initial concept hierarchy to save her effort from building the concept hierarchy from scratch. The initial concept hierarchy needs to be in good quality as a good starting point for the later interactive concept hierarchy construction (Chapter 6). Moreover, the quality of the initial concept hierarchy greatly affects efficiency in generating a concept hierarchy. We therefore need a strong automatic method to create good initial concept hierarchies.

This chapter presents metric-based concept hierarchy construction, a high performance automatic concept hierarchy construction framework [YC09b]. It is a comprehensive framework that incrementally clusters concepts based on an *hierarchy metric*, a score indicating semantic distance between concepts. It transforms the task of concept hierarchy construction into a multi-criterion optimization based on minimization of overall semantic distance of concepts, modeling of concept abstractness, and modelling of concept coherence.

Here we assume that all necessary concepts for building a concept hierarchy are available after concept extraction (Chapter 4). If some concept is missing, the algorithm will "make do" with what it has. Unlike most concept hierarchy construction algorithms which need to label new clusters, metric-based concept hierarchy construction bypass the cluster labeling and focuses on finding an optimal position for each concept. This may lead to some errors, but they can be later corrected interactively (Chapter 6).

This chapter begins with a discussion of the desirable properties in a concept hierarchy in Section 5.1. Section 5.2 introduces the terminologies being used in this chapter. Section 5.3 details the metric-based concept hierarchy construction framework. Section 5.4 presents the semantic feature functions. Section 5.5 elaborates 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.

5.1 Desirable Properties for A Concept hierarchy

A valid concept hierarchy should satisfy some commonsense requirements and preferably come with a few desirable properties. In this section, we explore the properties that would be desirable in a concept hierarchy. We study and learn from existing concept hierarchies that people use in daily life and conclude some common characteristics in them. We believe that a good concept hierarchy should represent a general and meaningful structure of the data. Generally, we consider a good concept hierarchy possessing the following desirable properties.

5.1.1 Minimum Semantic Distance and Minimum Evolution

One fundamental property of a good concept hierarchy is that concepts similar in meanings should be positioned close to each other. The closeness/similarity of concepts in meanings can be represented by a semantic distance. A good concept hierarchy should reflect the semantic proximity among concepts by organizing them into proper structures.

It is not trivial to assign accurate numeric scores to define semantic distances between concepts, however, we can at least expect them to be transitive, i.e., the nearer concepts have smaller distance scores. For example, we can distinguish the distance in meanings between "bird" and "wings" and the distance in meanings between "bird" and "university" - the former is closer and the latter is farther apart.

Positioning a concept at is correct positions means putting it in a correct neighborhood with correct parents, children, and siblings. Its semantic distance scores to its true neighbors

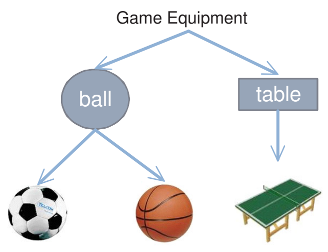


Figure 5.1: A concept hierarchy about "Game Equipment".

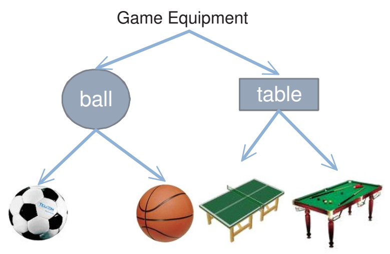


Figure 5.2: Add a new node "snooker table".

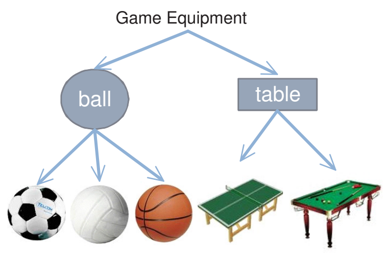


Figure 5.3: Add a new node "volleyball".

are small. If it is put into a wrong position, its semantic distance scores to the false neighbors are big. Therefore, a good concept hierarchy should guarantee that concepts related to the similar topics have small semantic distances among themselves and are put together. An optimal concept hierarchy should require minimum semantic distances among these concepts. This is a desirable property that we call "minimum semantic distance". A concept hierarchy with this property allows the users to have a good idea about why the concepts are put together. This greatly increases the interpretability of a constructed concept hierarchy.

On top of a local neighborhood, this property can also be applied to the entire concept hierarchy globally. When every part of a concept hierarchy satisfies "minimum semantic distance", it implies that every concept is placed in a neighborhood where it is most similar to, which further implies that the overall semantic distance among all concepts is minimal as well.

When a new concept being added into the concept hierarchy, there is an increase in the overall semantic distance of the entire concept hierarchy because it introduces several distances, which are non-negative and do not exist before, from itself to other nodes. However, how much this increase will be is decided by whether the new concept is inserted at the correct position. We observe that when a concept is in its correct position, it should give the least increase to the overall semantic distance in the concept hierarchy. Thus minimizing the overall semantic distance is equivalent to searching for the best possible position for a concept. Based on this principle, we can find the best positions for the concepts and organize them into ontologies. We call this property "minimum evolution". This idea is very

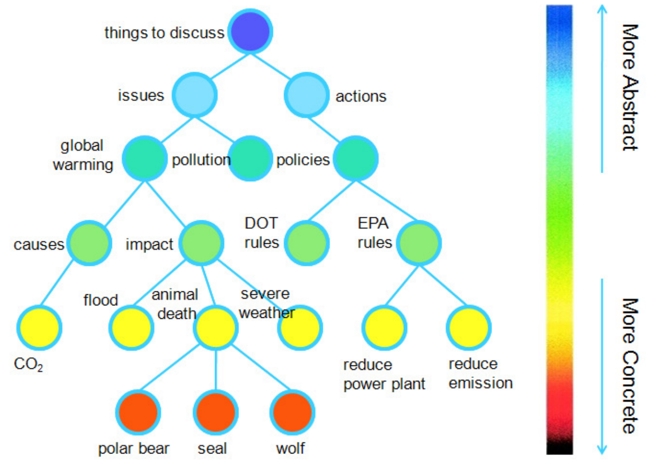


Figure 5.4: A good concept hierarchy.

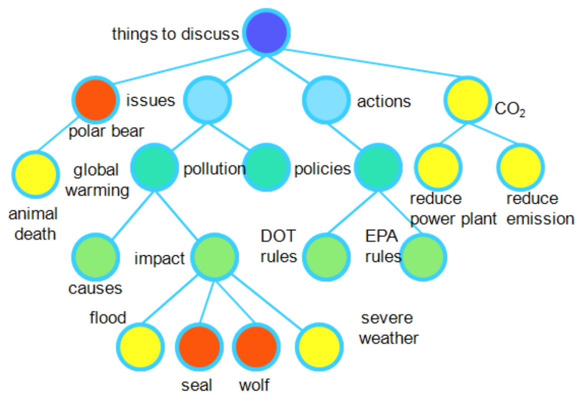


Figure 5.5: A problematic concept hierarchy.

similar to the minimum evolution tree selection criterion widely used in phylogeny [HP82] and minimum spanning tree in graph theory.

5.1.2 Abstractness

Figure 5.4 shows a concept hierarchy about "mercury pollution". It is a good concept hierarchy and was actually generated by one of the users in our user study (Chapter 7). In this concept hierarchy, concepts at the upper levels are more abstract, for example, "issues" and "actions"; while concepts at the lower levels are more concrete, for example, "polar bear" and "wolf". Figure 5.5 shows a concept hierarchy about the same domain. However, it is an problematic concept hierarchy in which some concepts are put into wrong positions. If we color-code the concepts by abstraction levels in Figure 5.4, we find that in Figure 5.5, concepts are misplaced at the wrong abstraction levels. This is the main reason why the concept hierarchy in Figure 5.5 seem problematic.

In general, for a good concept hierarchy, concrete concepts usually lay at the bottom of the hierarchy while abstract concepts often occupy the intermediate and the top levels. Concrete concepts often represent physical entities, such as "basketball" and "polar bear"; while abstract concepts, such as "issues" and "actions", 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 concept hierarchy construction. Therefore, a desirable property of a good

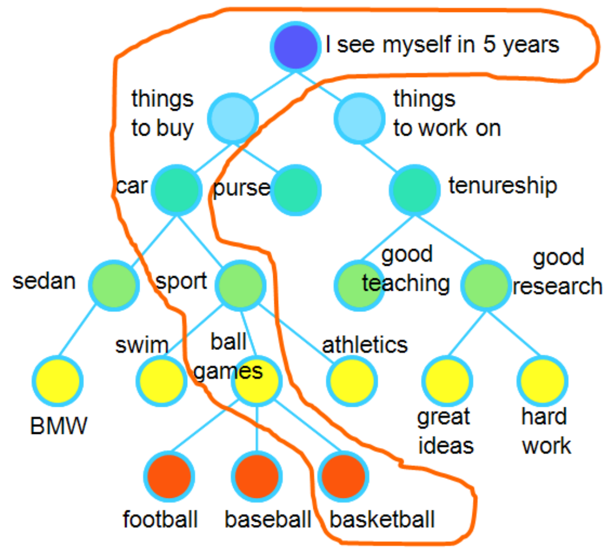


Figure 5.6: A problematic concept hierarchy: long distance incoherence.

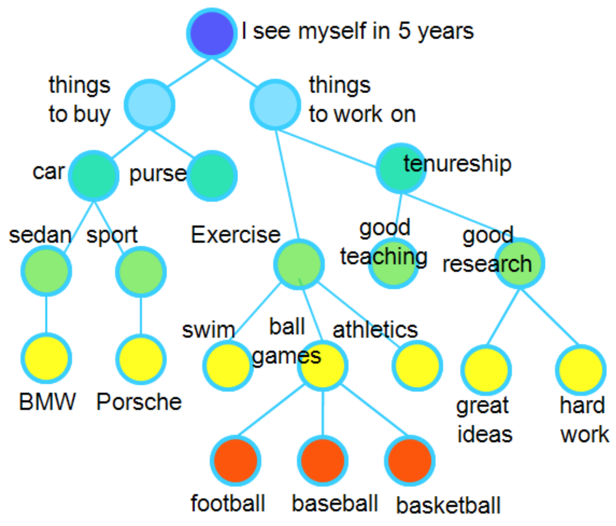


Figure 5.7: A good concept hierarchy.

concept hierarchy is that within a concept hierarchy, concepts at the different abstraction levels should be treated differently. We call this property "abstractness" property.

5.1.3 Long Distance Coherence

Most concept hierarchy construction algorithms take a bottom-up approach and only use local information to build ontologies. The bottom-up approaches focus on immediate relations, such as parent-child and sibling relations, but fail to verify the long distance relations, such as grandparent-grandchild and cousin-cousin relations. Therefore, it is common for them to generate incoherent concept chains.

This inconsistency is mainly caused by polysemies. For example, in Figure 5.6, under the concept "car", there are two nodes "sedan" and "sport"; they refer to subtypes of cars. Under "sport", there are "swim", "ball games", and "athletics"; they refer to "sporty games". When we evaluate these two groups of concepts separately, they both seem correct. However, without special constraints, the two groups are connected together because they share the same concept "sport". The resulting concept chain is obviously wrong due to the inconsistency in meanings along the path.

To avoid the long distance inconsistency, we require that for a good concept hierarchy, concepts on a “root-to-leaf” path should be coherent. That is, the overall semantic distances among the concepts along the path should be minimized. This property is called “long distance coherence”.

By applying this property, we can change the concept hierarchy in Figure 5.6 to Figure 5.7 (with a few more nodes), where the two concept groups are put in different paths and the concepts are coherent within each path.

5.1.4 Summary

The three desirable properties essentially introduce three optimization criteria that need to be minimized when we look for a best concept hierarchy. In Section 5.3, we discuss how to incorporate these optimization criteria into a unified concept hierarchy construction framework as objective functions.

With the desirable properties for ontologies in mind, we now describe the metric-based concept hierarchy construction framework. We start with a description of the definitions and terminologies used in this chapter.

5.2 Terminology: Concept hierarchy, Hierarchy Metric, and Information Function

To provide a theoretical formulation of the metric-based concept hierarchy construction framework, this section presents related terminologies used in this chapter.

Full Concept Hierarchy and Partial Concept Hierarchy

We take an incremental clustering approach to organize concepts into ontologies. The learning framework builds a concept hierarchy step by step by considering the concepts one after another and placing each concept at an optimal position in the concept hierarchy. The process starts with an initial concept hierarchy. The initial concept hierarchy can be empty or built either manually or by some simple techniques such as checking up in WordNet [Fel98]

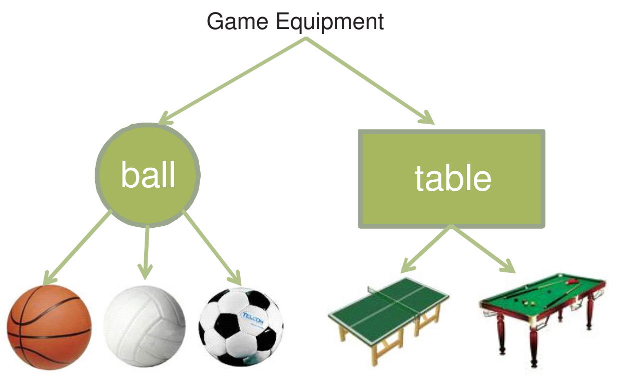


Figure 5.8: A full concept hierarchy. The concept set is {game equipment, ball, table, basketball, volleyball, football, pingpong table, snooker table}.

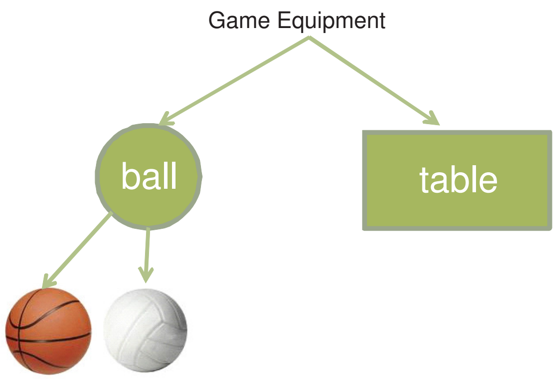


Figure 5.9: A partial concept hierarchy. The concept set is {game equipment, ball, table, basketball, volleyball, football, pingpong table, snooker table}. Concepts "basketball", "snooker table", and "pingpong table" are missing in the partial concept hierarchy.

or matching with lexico-syntactic patterns. We add concepts one by one to this initial concept hierarchy and obtain a series of "partial ontologies", each is formed after adding a new concept. When all the concepts in C are added into the concept hierarchy, the concept hierarchy is called a "full concept hierarchy". Below we give the definitions for "full concept hierarchy" and "partial concept hierarchy".

A *Full Concept Hierarchy* is a tree containing all the concepts in C. Formally,

$$T_{full} = (C, R|D)$$

s.t.

$$\forall c_x \in C, c_y \in C, c_x \neq c_y, \exists r(c_x, c_y) \in R.$$

A *partial concept hierarchy* is a tree containing only a subset of concepts in C. Formally,

$$T_{partial} = (C, R|D)$$

s.t.

$$\exists c_x \in C, c_y \in C, c_x \neq c_y, r(c_x, c_y) \notin R.$$

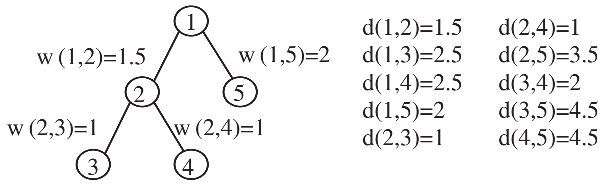


Figure 5.10: Illustration of hierarchy metric.

Figure 5.8 shows an example of a full concept hierarchy and Figure 5.9 shows an example of a partial concept hierarchy. The concept set C is $\{$*game equipment, ball, table, basketball, volleyball, football, pingpong table, snooker table*$\}$. The full concept hierarchy contains all the concepts in C while in the partial concept hierarchy, concepts "basketball", "snooker table", and "pingpong table" are missing.

Hierarchy Metric

We use *hierarchy metric* to define the semantic distance between two concepts. It is similar to *tree metric* used in graph theory [Tut01]. The difference is that hierarchy metric is especially designed for the metric-based concept hierarchy construction framework and is a function which can be applied on both leaf and non-leaf nodes; while tree metric can only be applied to leaf nodes.

Specifically, a *hierarchy metric* is a function $d : C \times C \longrightarrow \mathrm{R}_{+}$, where C is the set of concepts in a concept hierarchy T. The hierarchy metric d on a concept hierarchy T with edge weights w for any data pair (x, y) in the concept set C is defined as the sum of all edge weights along the path between the data pair:

$$d_{(T,w)}(x, y) = \sum_{e_{xy} \in P(x,y)} w(e_{xy}) \tag{5.1}$$

where $P(x, y)$ is the set of edges defining the path from concept c_x to concept c_y.

Figure 5.10 illustrates hierarchy metrics on a 5-node concept hierarchy. The five nodes in the concept hierarchy are labeled from 1 to 5, and the edge distances between two immediate nodes are also shown in the figure. The hierarchy metrics are calculated based on the edge distances. For example, the hierarchy metric between nodes 1 and 2 equals to their edge weight 1.5 because there is only one edge between these two nodes; the hierarchy metric between nodes 1 and 4 equals to 2.5, which is the sum of two edge weights, $w(1,2)+w(2,4) = 1.5 + 1 = 2.5$.

As a valid metric, a hierarchy metric d has to fulfill the following criteria:

- Non-negativity. $d(x, y) \geq 0$.
- Symmetricity. $d(x, y) = d(y, x)$.
- Equality condition. $d(x, y) = 0 \Leftrightarrow x = y$.
- Triangular inequality. $d(x, y) + d(y, z) \geq d(x, z)$.

Given the definition of hierarchy metric, triangular inequality falls into a special situation, where the sum of the hierarchy metrics for the ending points on two connected and non-overlapped paths equals the hierarchy metric for the ending points on the entire path. That is:

$$d(x, y) + d(y, z) = d(x, z).$$

In this dissertation research, we attempt to represent semantic distances using numeric scores, particularly, using hierarchy metrics. We formulate the hierarchy metric as a weighted combination of a set of underlying feature functions, and learn the weights for each feature from training data. The training data should come from reliable sources which can provide numerical representations of semantic distances through good examples. The reliable sources that we use include existing ontologies, such as WordNet and ODP, and direct instructions from the user who creates the concept hierarchy.

Information Functions

In Section 5.1, we discussed that an optimal concept hierarchy should minimize its overall semantic distance. In order to capture the overall semantic distance, we define an *information function* as the sum of the semantic distances for a set of concepts. The information function is a measure of information represented by a concept hierarchy or part of a concept hierarchy. We name it this way because the sum of semantic distances can be viewed as the amount of information or surprises in a concept hierarchy. The information function can also be considered as a cost function that we can minimize.

Formally, for a concept hierarchy $T(C, R)$, we define its information function $Info(T)$ as the sum of the hierarchy metrics among all pairs of concepts:

$$Info(T) = \sum_{c_x, c_y \in C, x<y} d(c_x, c_y) \tag{5.2}$$

where x, y are the indices for the two concepts c_x and c_y. The less sign "<" indicates that the index of c_x is less than that of c_y. The information function does not count the same pair of concepts twice.

For example, in Figure 5.10, the information function sums up the hierarchy metrics for all nodes in T: $Info(T) = d(1,2) + d(1,3) + d(1,4) + d(1,5) + d(2,3) + d(2,4) + d(2,5) + d(3,4) + d(3,5) + d(4,5) = 1.5 + 2.5 + 2.5 + 2 + 1 + 1 + 3.5 + 2 + 4.5 + 4.5 = 25$.

We can also define the information function over a subset of concept pairs in C. For example, we can define the information function over the concepts that are along the same root-to-leaf path $P_{s,t}$.

$$Info_{P_{s,t}} = \sum_{c_x, c_y \in P_{s,t}, x<y} d(c_x, c_y) \tag{5.3}$$

where x, y are the indices for the two concepts c_x and c_y, the less sign "<" indicates that the index of c_x is less than that of c_y. $P_{s,t}$ is the set of nodes along the path from s to t, where s is the root of P, and t is a leaf node at P.

For example, in Figure 5.10, node 2 and node 3 both belong to the path $P_{1,3}$, where node 1 is the root and node 3 is the leaf. The information function sums up the hierarchy metrics for all three nodes along the path $P_{1,3}$: $Info_{P_{1,3}} = d(1,2)+d(2,3)+d(1,3) = 1.5+1+2.5 = 5$.

5.3 The Metric-based Concept Hierarchy Construction Framework

This section presents the metric-based concept hierarchy construction framework. Based on the definitions of ontologies, hierarchy metric, and information functions, this section formulates the task of concept hierarchy construction as a optimization framework. It first shows how to estimate a hierarchy metric by learning the weights for a weighted combination of the underlying feature functions. Then based on the three desirable properties (Section 5.1), we formulate a multi-criterion optimization and solve the optimization problem by a greedy algorithm.

5.3.1 Estimating the Hierarchy Metric

A hierarchy metric defines the semantic distance between two concepts as the sum of edge weights along a path between them (Section 5.2). This definition is based on the assumption that the edge weights are known. However, in general this assumption is not true and we need to learn the pair-wise hierarchy metrics from the training data. In this section, we demonstrate how to estimate the hierarchy metrics.

Learning a good hierarchy metric is important for the metric-based concept hierarchy construction framework. We adopt a general definition of distance for hierarchy metric and assume that there are some underlying feature functions which measure semantic distances from various aspects. Particularly, we represent a hierarchy metric as a Mahalanobis distance function [Mah36]. The hierarchy metric for concepts (c_x, c_y) is formulated as:

$$d_{c_x,c_y} = \sqrt{\Phi(c_x, c_y)^T W^{-1} \Phi(c_x, x_y)} \tag{5.4}$$

where $\Phi(c_x, c_y)$ represents a set of pairwise underlying feature functions for concepts c_x and c_y, where each feature function is $\phi_k : (c_x, c_y)$ with k=1,...,$|\Phi|$. W is a weight matrix, whose diagonal values weigh the underlying feature functions.

Mahalanobis distance is a general parametric function widely used in distance metric learning [Yan06]. It measures the dissimilarity between two random vectors of the same distribution with a covariance matrix W. The covariance matrix W scales the data points from their original values by $W^{1/2}$. When W is the identityl matrix, Mahalanobis distance reduces to Euclidean distance. When only diagonal values of W are taken into account, W is equivalent to assigning *weights* to different axes in the random vectors. Hence it can be viewed as a weight matrix for weighing different feature functions if the random vectors represent the feature vectors.

Note that a hierarchy metric is still a distance metric. One important characteristic of a valid distance metric is that it must represent valid clustering partitions, which means that the clustering partitions represented by the distance metric 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 W to be positive semi-definite (PSD) [Bha06]: $W \succeq 0$.

We choose Mahalanobis distance as the form of hierarchy metric for two main reasons. (1) It is in a parametric form which allows us to learn a distance function by supervised learning. Hence it provides an opportunity to assign different weights for each type of semantic features. (2) When W is properly constrained to be positive semi-definite, a Mahalanobis-formatted distance will be guaranteed to satisfy non-negativity and triangle inequality. Therefore, in this research, we use the Mahalanobis-formulated distance to measure the semantic distance between two concepts by assigning adaptive weights to different underlying feature functions. As long as these two conditions are satisfied, one may use other forms of distance functions to represent a hierarchy matric. For more information on distance metric learning, see [Yan06].

In Equation 5.4, the feature functions $\Phi(c_x, c_y)$ and their values are generated from the data (both training and test data) itself. It is the weight matrix W that we need to learn from the training data and then based on it to estimate the hierarchy metric for test data. Therefore learning a hierarchy metric becomes two subproblems - getting good features and learning the weight matrix from the training data.

A feature functions $\Phi(c_x, c_y)$ can be any function which measures semantic relations between two concepts c_x and c_y. For example, it can be a contextual feature, co-occurrence, a syntactic dependency feature, a lexico-syntactic pattern, word length difference, or overlap of two concept's definitions. We use heterogonous feature functions to evaluate the semantic relation between two concepts and aim to capture a wide range of characteristics for their semantic relations. Section 5.4 gives more details on the feature functions.

We can estimate W by minimizing the expected loss between the hierarchy metrics d generated from the training data and the expected value $\hat{d}$. Theoretically, the expected loss (or the risk) can be represented as :

$$R(\hat{\mathbf{d}}) = \int \Delta(\mathbf{d}, \hat{\mathbf{d}}) p(\mathbf{c}, \mathbf{d}) d\mathbf{c} d\mathbf{d} \tag{5.5}$$

where $\hat{\mathbf{d}}$ is the expected hierarchy metric, $\mathbf{d}$ is the observed hierarchy metric in the training data, $\mathbf{x}$ is the concepts, and $p(\mathbf{c}, \mathbf{d})$ is the joint distribution of concepts and their hierarchy metrics.

We choose to minimize the expected loss through minimizing the squared errors. Thus the loss function Δ in Equation 5.5 is defined as the squared error loss of two hierarchy metrics. In the training data, a hierarchy 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.

Using minimization of squared error and constraining the weight matrix W to be positive semi-definite, the optimization function for the parameter estimation can be formulated as:

$$\min_W \sum_{x=1}^{|\mathbf{C}|} \sum_{y=1}^{|\mathbf{C}|} \left(d_{xy} - \sqrt{\Phi(c_x, c_y)^T W^{-1} \Phi(c_x, c_y)} \right)^2 \tag{5.6}$$

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

where d_{xy} is the abbreviation of d_{c_x,c_y}, $\Phi(c_x, c_y)$ represents a set of pairwise underlying feature functions; W is the weight matrix, which weighs the underlying feature functions.

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

5.3.2 The Minimum Evolution Objective

Section 5.1 described three desirable properties that a good concept hierarchy should possess. In this section and the following sections, we apply these properties to formally define the optimization criteria for concept hierarchy construction. We start with the minimum evolution property in this section.

Metric-based concept hierarchy construction takes an incremental clustering approach to organize concepts into ontologies. The learning framework builds a concept hierarchy step by step by considering the concepts one by one and placing each concept at an optimal position

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

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

in the concept hierarchy. Based on the minimum evolution property presented in Section 5.1, the algorithm inserts each concept into the concept hierarchy so that the insertion minimizes the overall semantic distances. The algorithm searches through all the possible positions for the new concept and evaluates the resulting overall semantic distance due to this insertion and select the one minimizes the overall semantic distance at each step.

Moreover, the minimum evolution property suggests to avoid dramatic structure changes when a new node is added. We thus define that the goal of concept hierarchy construction is to find an optimal full concept hierarchy $\hat{\mathrm{T}}$ such that the information changes are the least from the initial concept hierarchy T^0, i.e., to find:

$$\hat{T} = \arg\min_{T'} \Delta Info(T^0, T') \tag{5.7}$$

where T' is a full concept hierarchy, whose concept set equals to C.

To find the optimal solution for Equation 5.7, 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 concept hierarchy is always C, the only unknown part is $\hat{R}$. Thus, Equation 5.7 can be equivalently transformed into:

$$\hat{R} = \arg\min_{R'} \Delta Info(T'(C, R'), T^0(S^0, R^0)) \tag{5.8}$$

where S^0 is the initial concept set, R^0 is the initial relation set.

Every insertion of concepts produces a new partial concept hierarchy. Therefore the optimal partial concept hierarchy at each insertion step is the one that gives the least information change. At each insertion, after adding the n^{th} concept, the best current concept hierarchy T^n is one that introduces the least change of information from the previous concept hierarchy T^{n-1}:

$$T^n = \arg\min_{T'} \Delta Info(T^{n-1}, T') \tag{5.9}$$

where the *information change function* is the absolute difference between the information of these two ontologies:

$$\Delta Info(T^{n-1}, T') = |Info(T^{n-1}) - Info(T')|. \tag{5.10}$$

The updating function for the set of relations R^n after the n^{th} concept z is inserted can be calculated as:

$$\begin{aligned} R^{(n)} &= \arg\min_{R'} \Delta Info(T^n, T^{n-1}) \\ &= \arg\min_{R'} \Delta Info(T(S^{n-1} \cup \{c_z\}, R'), T(S^{n-1}, R^{n-1})) \end{aligned} \tag{5.11}$$

By plugging in the definitions of the information change function information function $Info(.)$ (Equation 5.2), the updating function becomes the minimization of the absolute difference in the information for the ontologies with and without the n^{th} concept:

$$R^{(n)} = \arg\min_{R'} | \sum_{c_x, c_y \in S^{n-1} \cup \{c_z\}} d(c_x, c_y) - \sum_{c_x, c_y \in S^{n-1}} d(c_x, c_y)| \tag{5.12}$$

where $d_{(c_x, c_y)}$ is the hierarchy metric between c_x and c_y.

After transforming the absolute function into constraints, we can formulate Equation 5.12 into a minimization problem as follows:

$$\min u \tag{5.13}$$

$$\text{subject to } u \leq \sum_{c_x, c_y \in S^n \cup \{c_z\}} d(c_x, c_y) - \sum_{c_x, c_y \in S^n} d(c_x, c_y) \tag{5.14}$$

$$u \leq \sum_{c_x, c_y \in S^n} d(c_x, c_y) - \sum_{c_x, c_y \in S^n \cup \{c_z\}} d(c_x, c_y) \tag{5.15}$$

$$x < y \tag{5.16}$$

where the first two constraints (Equation 5.14 and Equation 5.15) guarantee that the information changes are bounded by the absolute difference between T^n and T^{n-1}. The third constraint (Equation 5.16) defines the order of the concepts so that we do not compute the pair-wise distances twice.

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

To look for the best concept hierarchy according to this objective, we need to exhaustively search for the best position among all the possible positions in a concept hierarchy for every new concept. This process is not efficient. We need to find good constraints to limit the search space. The constraints need to be reasonable and obey the desirable properties for a good concept hierarchy. The other two desirable properties, "abstractness" and "long distance coherence", nicely provide such constraints to reduce the search space in minimum evolution algorithm. Since this property requires that concepts at a horizontal level to follow similar characteristics, this property is equivalent to posting many *horizontal* constraints to a concept hierarchy. It not only produces a more sensible concept hierarchy since it requires the learning algorithm to obey concept abstractness, but also reduces the possible search space for the optimal concept positions hence improves the efficiency. Similarly, the coherence property emphasizes on the "root-to-leaf" path, and is equivalent to posting many *vertical* constraints to a concept hierarchy. We describe the two constraints/objectives in the following two sections.

5.3.3 The Abstractness Objective

Metric-based concept hierarchy construction addresses concept abstractness in ontologies. Unlike most prior research (Chapter 2), 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 concept hierarchy construction supports different feature functions at different abstraction levels.

According to the abstractness property (Section 5.1), we formulate the learning framework such that concepts on the same abstraction level share the same weighted combinations of the feature functions to derive hierarchy metrics; while concepts at different abstraction levels do not share the same weights for their hierarchy metrics because they should have different characteristics.

Particularly, we model concept abstractness explicitly by learning separate weight matrices for concepts at different abstraction levels. Suppose L_i is the subset of concepts lying at the i^{th} level of a concept hierarchy T, and i is the index of the levels in a concept hierarchy. 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. To demonstrate, in Figure 5.10, node 1 is at level L_1, node 2 and node 5 are at level L_2.

Based on the abstractness property and the above notations about abstraction levels, we formulate the abstractness property as follows:

$$\forall c_x \in L_i, c_y \in L_j, |i-j| > 1, \tag{5.17}$$

$$d(c_x, c'_x) = \sqrt{\Phi(c_x, c'_x)^T W_{L_i}^{-1} \Phi(c_x, c'_x)}; \tag{5.18}$$

$$d(c_y, c'_y) = \sqrt{\Phi(c_y, c'_y)^T W_{L_j}^{-1} \Phi(c_y, c'_y)}; \tag{5.19}$$

$$W_{L_i} \neq W_{L_j} \tag{5.20}$$

where c'_x is one of the immediate neighbors (parents or children) of c_x, $d(c_x, c'_x)$ is the hierarchy metric which measures the semantic distance between c_x and its immediate neighbor c'_x, and $\Phi(c_x, c'_x)$ is the set of feature functions. c'_y, $d(c_y, c'_y)$, and $\Phi(c_y, c'_y)$ are defined for concept c_y similarly. W_{L_i} and W_{L_j} are the weight matrices used for concepts at abstraction levels L_i and L_j respectively. Since we only consider the cases when c_x and c_y are not immediate neighbors, L_i and L_j are not next to each other and their indices $|i-j| > 1$. In the case that c_x and c_y are immediate neighbors, we do not force this property to be true since they may share the same hierarchy metrics $d(c_x, c_y)$.

An abstraction level L_i is characterized by its own weight matrix W_{L_i}. Using minimization of squared error and constraining the weight matrix W to be positive semi-definite, the parameter estimation for W at abstraction level L_i is:

$$W_{L_i} = \min_W \sum_{x=1}^{|L_i|} \sum_{x'=1}^{|N(c_x)|} \left(d_{xx'} - \sqrt{\Phi(c_x, c'_x)^T W^{-1} \Phi(c_x, c'_x)} \right)^2 \tag{5.21}$$

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

where c'_x is an immediate neighbor to c_x, $|N(c_x)|$ is the number of concepts in the c_x's neighborhood $N(c_x)$, $d_{xx'}$ is the abbreviation for $d(c_x, c'_x)$.

This minimization follows the abstractness property; hence we call it the *abstractness objective.*

Since modeling of concept abstractness is done by approximating some characteristics for each abstraction level as a weighted combination 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 various features.

5.3.4 The Coherence Objective

Metric-based concept hierarchy construction addresses concept coherence in long distance. Along a path from the root to a leaf in a concept hierarchy, the framework requires all the concepts to be about the same topic. They need to be coherent no matter how far away two concepts are apart in this path. This requires the sum of the semantic distances in the path to be as small as possible.

When a new concept c_x is added into a concept hierarchy T, the long distance coherence property (Section 5.2) requires that the optimal root-to-leaf path $\hat{P}$ containing c_x should satisfy the following condition:

$$\hat{P} = \arg\min_{P'} Info'_P \tag{5.22}$$

where $\hat{P}$ is an optimal path which c_x eventually goes to. P' is one of the possible paths that c_x can be added into T.

Recall that we define the information function for concepts in a root-to-leaf path in Equation 5.3 as:

$$Info_{P_{s,t}} = \sum_{c_x, c_y \in P_{s,t}, x<y} d(c_x, c_y)$$

where $P_{s,t}$ is the set of nodes along the path from concept s to concept t. By plugging in the definition, when a new concept c_z arrives at a path P, the information function for this particular path should be minimized as follows:

$$\hat{P_{c_z}} = \arg\min_{P'_{c_z}} \sum_{c_x, c_y \in P'_{c_z}, x<y} d(c_x, c_y) \tag{5.23}$$

where P_{c_z} is a root-to-leaf path including c_z, $x < y$ defines the order of the concepts so we only compute a pair-wise distance between two concepts once.

This minimization follows the long distance relation property; hence we call it the *long distance objective.*

5.3.5 The Multi-Criterion Optimization Algorithm

We have presented three optimization objectives - *minimum evolution*, *abstractness*, and *coherence* objectives. It is necessary to unify them into a single optimization framework and seek the optimal solution for this framework. This section presents the unified objective function and the optimization algorithm.

As an incremental concept hierarchy construction framework, metric-based concept hierarchy construction handles the insertion of concepts one by one. At each insertion, a new concept c_z is added into the existing partial concept hierarchy, which produces a new partial concept hierarchy and a new set of relations $R(c_z, .)$. The new set of relations decide where to put this concept. The evaluation of the best position depends on evaluating the semantic distances of concept c_z and other concepts in the concept hierarchy. Particularly, we need to know the hierarchy metrics $d(c_z, .)$ between concept c_z and its immediate neighbors. According to the abstractness property, inserting c_z to different abstraction levels yields different hierarchy metrics between c_z and other concepts. The prediction of the hierarchy metric is done through estimation of the weight matrices at all different abstraction levels. The weight matrices W_{L_i} are estimated as in Equation 5.21.

Based on the learned weight matrices, the hierarchy metrics for c_z can be predicted as:

$$\forall i = 1, ..., |depth(T)|,$$

$$d_{L_i}(c_z, .) = \sqrt{\Phi(c_z, .)^T W_{L_i}^{-1} \Phi(c_z,)} \tag{5.24}$$

The minimum evolution and coherence objectives decide the best position that a new concept should be added into the concept hierarchy. Particularly, c_z is tried at every possible position in the current partial concept hierarchy T^n as either a parent or a child to an existing node. The minimum evolution objective picks the optimal position to insert c_z, which minimizes both the overall semantic distances and the information change from the previous partial concept hierarchy. The coherence objective watches for a root-to-leaf path P'_{c_z} where c_z is inserted into and requires that the semantic distances of the path is minimized. To optimize the “minimum evolution” and “coherence” objectives, we introduce a variable $\lambda \in [0, 1]$ to control the contribution from each objective. The multi-criterion optimization function is formulated as follows:

```
ALGORITHM 5.3.5: Multi-Criteria Optimization.
foreach $i = 1, ..., |depth(T_{training})|$
    $W_{L_i} = \min_W \sum_{x=1}^{|L_i|} \sum_{y=1}^{|N(c_{training_x})|} \left(d_{xy} - \sqrt{\Phi(c_{training_x}, c_{training_y})^T W_{L_i}^{-1} \Phi(c_{training_x}, c_{training_y})}\right)^2$;
foreach $c_z \in C \setminus S$
    $S \leftarrow S \cup \{c_z\}$;
    foreach $i = 1, ..., |depth(T)|$
        if $W_{L_i} \succeq 0$
            $d_{L_i}(c_z, .) = \sqrt{\Phi(c_z, .)^T W_{L_i}^{-1} \Phi(c_z, )}$;
    $R \leftarrow R \cup \{\arg\min_{R_{(c_z,.)}} (\lambda u + (1-\lambda) v)\}$;
Output $T(S, R)$
```

Figure 5.11: The algorithm for multi-criteria optimization.

$$\min \lambda u + (1-\lambda) v \tag{5.25}$$

$$\text{subject to } u \le \sum_{c_x, c_y \in S^n \cup \{c_z\}} d(c_x, c_y) - \sum_{c_x, c_y \in S^n} d(c_x, c_y), \tag{5.26}$$

$$u \le \sum_{c_x, c_y \in S^n} d(c_x, c_y) - \sum_{c_x, c_y \in S^n \cup \{c_z\}} d(c_x, c_y), \tag{5.27}$$

$$v = \sum_{c_j, c_k \in P'_{c_z}, j<k} d(c_j, c_k), \tag{5.28}$$

$$x < y, \tag{5.29}$$

$$0 \le \lambda \le 1. \tag{5.30}$$

where u represents the "minimum evolution" objective, and v represents the "coherence" objective, $x < y$ defines the order of the concepts so we only compute a pair-wise distance between two concepts once, P'_{c_z} is a root-to-leaf path where c_z is inserted into, and S^n is the concept set for the n^{th} partial concept hierarchy.

The above optimization can be solved by a greedy algorithm. Moreover, the optimal solution can be indicated by an optimal set of relations $R(c_z, .)$. Each time when a new concept arrives, the algorithm first estimates its hierarchy metrics, and then based on that to find the optimal position for the multi-criterion optimization. It is outlined as shown in Algorithm 5.3.5.

This greedy algorithm presents a general incremental procedure to construct ontologies. By minimizing the concept hierarchy structure changes, modeling concept abstractness, and modelling long distance coherence at each step, it finds the optimal position of each concept in a concept hierarchy.

During this incremental process, every concept is added into the hierarchy one by one. Suppose there are N concepts in total to be added. Among these N concepts, assume m of them are already in the concept hierarchy, a new concept will be added into this size-m concept hierarchy. We need to put this new concept at some temporary positions within the current concept hierarchy, then compute the new overall semantic distance for the minimum evolution objective and path semantic distance for the coherence objective. The temporary positions that we try are either a dummy parent or a dummy child to an existing node in the current hierarchy. For each new concept, this process yields $2m$ operations. Moreover, for each temporary position, when calculating semantic distances for the minimum evolution objective, the time complexity is $O(m)$. This is because that we need to compute the distance between the new concept at the temporary position to all existing m nodes in the hierarchy. Note that for the pairwise distances between existing nodes in the hierarchy, their distances should already have been calculated in the previous iterations and therefore no additional computational cost remains for this iteration. The abstractness objective is used to estimate semantic distances. Its computation mainly happens during training phrase thus does not contribute time costs to the hierachy construction process. For the concept coherence objective, it needs to compute the path semantic distance for each path. However, all the pairwise distances have already been calculated either in the previous iterations or in the step to calculate minimum evolution objective, therefore, coherence objective does not introduce much overhead cost, its time complexity is $O(1)$. Therefore, for each newly added concept, the algorithm yields a time complexity of $O(2m^2)$: distances to existing nodes - m, and trying different positions - $2m$.

Since we grow the concept hierarchy by adding more and more nodes from scratch, m increases from 1 to $N-1$. Therefore the overall time complxity is $O(2*1^2+2*2^2+2*3^2+ ... + 2*(N-1)^2) = O(\sum_i^{N-1} 2*i^2) = O(\frac{1}{3}(N-1)N(2N-1)) = O(N^3)$. Thus considering all three objectives, the big O notation for this greedy algorithm is $O(N^3)$.

The order in which concepts are added into the concept hierarchy may affect the structure of the final concept hierarchy. Currently we insert concepts in an arbitrary order for multiple random restarts. The best resulting concept hierarchy is then selected as the final concept hierarchy. In practice, we perform 5 to 15 random restarts with different random S^0 and pick the best resulting T', which gives the least information change among all possible candidates.

5.4 The Features

The features used in this dissertation research are indicators of semantic relations between two concepts [YC09a]. Given two concepts c_x and c_y, a feature is defined as a function which generates a numeric score $\phi(c_x, c_y) \in \Re$. A weighted combination of these features functions in the learning algorithm make it possible to learn and adjust the weights for each feature.

In this dissertation research, we concentrate on six commonly-used types of features for measuring semantic distances. The types of features include *contextual*, *co-occurrence*, *syntactic dependency*, *lexico-syntactic patterns*, and *other* features.

All feature values are normalized and lie in the range of [0,1]. The higher a feature value, the more similar the concept pair is according to this feature.

5.4.1 Contextual Features

The first type of features employ *context* to measure the semantic distance between two concepts. As stated in the Distributional Hypothesis [Har54], words appearing in similar contexts tend to be similar. Therefore, word meanings can be inferred from and represented by contexts. We hence develop the following contextual features:

Google Global Context KL-Divergence

This feature function measures the Kullback-Leibler divergence (KL divergence) [SMTC05] between language models associated with two input concepts c_x and c_y.

We build a Web-based auxiliary dataset to assist generating this feature. Besides the document collection for which we construct the concept hierarchy, an auxiliary dataset provides additional context for the concepts. Moreover, when only concepts but no documents are given for concept hierarchy construction, the auxiliary datasets become the only source to obtain contextual information for the concepts.

The Google auxiliary dataset collects the top 1,000 documents returned by the Google search engine when querying Google with a concept or a concept pair. The Web documents in the auxiliary dataset are split into sentences, and parsed by a POS tagger[3], a semantic role tagger (ASSERT[4]), and a syntactic parser (Minipar[5]). This dataset is used to generate not only this feature, but also other features, including *Google local context KL-divergence, document PMI, sentence PMI, Minipar syntactic distance, modifier/object/subject/verb overlap*, *hypernym/sibling/part-of pattern-based features*, and *Google PMI.*

The global context of a concept is considered as the Web documents returned for it. We build the global context for each concept into a unigram language model. KL divergence between two language models associated with the two input concepts is used as the output value for this feature function. Particularly, if the language model for concept c_x is p, for c_y is q, the KL divergence can be calculated as:

$$D_{KL}(p||q) = \sum_i p_i log_2 \frac{p_i}{q_i}$$

where i is the index for the i^{th} word in p, p_i is this word's frequency in p, and q_i is its frequency in q. If $p_i = 0$ or $q_i = 0$, Dirichlet smoothing [ZL04] is applied.

Wikipedia Global Context KL-Divergence

Similar to the Google Global Context KL-Divergence, this feature function calculates the KL divergence between two unigram language models associated with the two input concepts. The only difference is that the Google auxiliary dataset is substituted by a Wikipedia auxiliary dataset.

To create the Wikipedia auxiliary dataset, we download the entire Wikipedia document collection and index it by the Indri search engine[6]. The Wikipedia auxiliary dataset collects the top 100 documents returned by Indri when querying the index with a concept. If the number of returned documents is less than 100, we just keep all the returned ones. We then build unigram language models for the concepts and calculate the KL divergence between

[3]http://nlp.stanford.edu/software/tagger.shtml.
[4]http://cemantix.org/assert.
[5]http://www.cs.ualberta.ca/lindek/minipar.htm.
[6]http://www.lemurproject.org/indri/.

the models in a way similar to the *Google global context KL-divergence.*

Google Local Context KL-Divergence

This feature function uses the Google auxiliary dataset to extract local contexts for concepts and compares their local contexts. We define the local context of a concept c_x is the left two and the right two words surrounding it. Among all the documents containing c_x in the Google auxiliary dataset, the surrounding words around c_x are collected and built into a unigram language model for the concept. Similarly, we can derive a unigram language model for the other concept c_y using its surrounding words. This feature function outputs the KL divergence of the two unigram models.

Wikipedia Local Context KL-Divergence

Similar to the *Google local context KL-divergence*, this feature function uses the local context of concepts, i.e., the left two and the right two words surrounding the concepts, to calculate a KL-divergence between two concepts. The only difference is that we use the Wikipedia auxiliary dataset as the document collection for extracting the local contexts.

5.4.2 Co-occurrence Features

The second type of features use *co-occurrence* information of two concepts. Specifically, we measure the co-occurrence of two concepts c_x and c_y by their point-wise mutual information (PMI):

$$pmi(c_x, c_y) = log\frac{Count(c_x, c_y)}{Count(c_x)Count(c_y)} \tag{5.31}$$

where the $Count()$ function can be defined at different levels of text granularity. For example, it can be the number of documents containing one of the concepts or containing both concepts. It can also be the number of sentences containing c_x or c_y alone or containing both. Based on different definitions of $Count(.)$, we have the following *co-occurrence* features:

Document-level PMI

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

Sentence-level PMI

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

Google PMI

This feature function measures Web-based PMI according to the total number of search results returned by Google for a concept or a pair of concepts. $Count(.)$ is defined as n for concept c_x as in the line of "Results 1-10 of about n for c_x" appearing on the first page of Google search results using c_x as the search query. For a concept pair, we query Google by using a concatenation of the two concepts.

5.4.3 Syntactic Dependency Features

The third type of features employ *syntactic dependency* analysis. We use Minipar[7] to generate syntactic parse trees and based on them to induce a feature called "Minipar syntactic distance". We also use ASSERT (Automatic Statistical SEmantic Role Tagger)[8] to produce semantic role labels for each candidate predicate in sentences. The semantic similarity between two concepts are measured based on how much they overlap in terms of their neighboring semantic roles. We have the following syntactic dependency features:

Minipar Syntactic Distance

This feature function measures the number of edges between two concepts c_x and c_y in a syntactic parse tree. It is the averaged length of the shortest syntactic paths (in the first syntactic parse tree returned by Minipar) between two concepts. In particular, for each sentence s_i that contains both c_x and c_y, we use Minipar to generate a syntactic parse tree

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

[8]http://cemantix.org/assert.

t_{s_i} for s_i; c_x and c_y appear as two nodes in t_{s_i}. Let d_i be the shortest edge distance, i.e., the number of edges, between c_x and c_y in t_{s_i}. Then this feature function outputs an averaged value of d_i for all i where sentences s_i containing both c_x and c_y. The sentences are from both the original document collection and the Google auxiliary dataset.

Modifier Overlap

This feature function measures the number of overlaps between modifiers for each of the two concepts. The modifiers are identified based on the tags produced by Minipar. Suppose a sentence s_i contains c_x. We parse the sentence with Minipar. Within a parse tree, we define a *modifier* to a concept c_x to be the Adjective (ADJ) or Noun (NN, NNS, NNP, or NNPS) on the left hand side of c_x but still within the scope of c_x's parent node.

For example, suppose concept c_x is "federal rule", a Minipar parsed tree for the sentence "An EPA proposed federal rule applies." is:

> [S [NP [[DET An] [NP [NN EPA] [ADJ proposed] [ADJ federal] [NN rule]]] [VP applies].]

The parent node for "federal rule" is "NP". Within its scope and on the left hand side of "federal rule", "EPA" is tagged as NN and "proposed" is tagged as ADJ; both words are considered as the modifier for "federal rule". Thus the modifier in this sentence for "federal rule" is "EPA proposed".

We gather all unique modifiers for c_x in all sentences in the original document collection and the Google auxiliary dataset. Thus we have a list of modifiers for c_x. Similarly, we can find the modifiers for c_y. The number of overlaps between the two lists of modifiers is the output value for this feature function.

Object Overlap

This feature function measures the number of overlaps between objects in sentences with two concepts c_x and c_y as the sentences' subjects. The objects and other semantic roles (such as subjects or verbs) are recognized by ASSERT. Suppose a sentence s_i contains c_x. We parse the sentence with ASSERT. The object for a concept c_x is labeled as "OBJ" (or "ARG1" in ASSERT's codes) in the parsed sentence.

For example, suppose c_x is "the EPA", and the sentence "The EPA should require power plants to cut mercury pollution by 90% by 2008" is labeled as follows:

[ARG0 The EPA] [ARGM-MOD should] [TARGET require] [ARG1 power plants] [ARGM-PNC to cut mercury pollution by 90% by 2008]

In the labeled sentence, "the EPA" is the subject and is labeled as "ARG0" in ASSERT's codes. The object is "power plants", which is labeled as "ARG1" in ASSERT's codes. Thus the object for "the EPA" is "power plants" in this sentence.

We gather all the unique objects for c_x in all sentences in the original document collection and the Google auxiliary dataset. Similarly, we can obtain a list of unique objects for c_y. We then compare the number of overlaps between the two lists of objects for these two concepts and output the resulting value for this feature function.

Subject Overlap

This feature function measures the number of overlaps between subjects in sentences with c_x or c_y as the sentences' object. We also use ASSERT to identify the subjects for two concepts in a similar manner as we calculate the value for *object overlap*. The only difference is that c_x and c_y are now labeled as OBJ (or "ARG1" in ASSERT's codes) and the subjects are labeled as SUBJ (or "ARG0" in ASSERT's codes). We gather all the unique objects for c_x in all sentences in the original document collection and the Google auxiliary dataset.

Verb Overlap

This feature function measures the number of overlaps between verbs for c_x and c_y in sentences containing them. We use ASSERT to identify the verbs for the two concepts in a similar manner as we calculate the value for *object overlap* and *subject overlap*. The only difference is that c_x and c_y are either labeled as SUBJ (or "ARG0" in ASSERT's codes) or OBJ (or "ARG1" in ASSERT's codes) and the verb are labeled as TARGET by ASSERT. We gather all the unique objects for c_x in all sentences in the original document collection and the Google auxiliary dataset.

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

Table 5.1: Lists of lexico-syntactic patterns. In a hypernym pattern, NP_x indicates the slot for a hyponym concept and NP_y indicates the slot for the hypernym concept. In a part-of pattern, NP_x indicates the slot for a meronym concept and NP_y indicates the slot for the coordinate concept. In a sibling pattern, both NP_x and NP_y indicate the slots for two sibling concepts. ? indicates optional.

5.4.4 Lexico-Syntactic Patterns

The fourth type of features are *lexico-syntactic patterns*. Table 5.4.4 lists all patterns used in this dissertation research. For a hypernym pattern, NP_x indicates the slot for a hyponym concept and NP_y indicates the slot for a hypernym concept. For a part-of pattern, NP_x indicates the slot for a meronym concept and NP_y indicates the slot for a coordinate concept. For a sibling pattern, both NP_x and NP_y indicate the slots for two sibling concepts. The pattern-based features include:

Hypernym Patterns

This feature function is based on the patterns proposed by [Hea92] and [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 the original document collection or the Google auxiliary dataset, 0 otherwise.

Sibling Patterns

This feature function contains patterns indicating a sibling relation. They are 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 the original document collection or the Google auxiliary dataset, 0 otherwise.

Part-of Patterns

This feature function is based on patterns proposed by [GBM03] and [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 the original document collection or the Google auxiliary dataset, 0 otherwise.

5.4.5 Other Features

We also use simple features such as word length or definition differences to measure the semantic distance between two concepts. We call them "other features". They include:

Word Length Difference

Word length has been shown impact on how people recognize and memorize words. It is influenced by a word's abstractness. We measure the word length difference between two concepts (including both words and phrases) as one of the semantic distance measures. This feature function returns the length difference (excluding white spaces) between two concepts c_x and c_y. For example, if c_x is "basketball", and c_y is "sport", their word length difference is $10 - 5 = 5$.

Definition Overlap

This feature function measures how similar the definitions for two concepts c_x and c_y are. In particular, we submit a query for c_x to Google to get its definitions. The query is in the format of "define:c_x". For example, if c_x is "government", we form the query as "define:government" and submit it to Google. Google returns a page containing a list of entries, each explaining the meaning of the word. Here we only show two definitions for "government" in this example as follows:

- the body with the power to make and/or enforce laws to control a country, land area, people or organization. [en.wiktionary.org/wiki/government, 10/9/2011]

- a political institution that decides, regulates, controls, and enforces public policy. [www.teogathalaw.com/tax-glossary.php, 10/9/2011]

Similarly, we can get the definitions for c_y. After removing stop words, and applying stemming, we compare the number of word overlaps between the definitions for the two concepts and normalize the value by the length of the definitions.

5.4.6 Summary

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 the learning framework allows us to use all of them. This more general framework has the potential to learn more complex ontologies than prior studies.

5.5 Evaluation

This section presents the evaluation for the metric-based concept hierarchy construction framework. In Section 5.5.1 we describe our training and test sets, measurement, and baseline systems. We then present various experiments in the following sections. Section 5.5.2 tests the performance of the metric-based concept hierarchy construction framework for noun concept hierarchy construction and also compares our system with three baseline systems. Section 5.5.3 reports the effect of concept abstractness. Section 5.5.4 presents the experiments on concept coherence. Section 5.5.5 studies the impact of individual features on different relations. Section 5.5.6 studies the interaction of individual features and abstraction levels. Section 5.5.7 studies the stability of concept hierarchies constructed under slight document set changes.

5.5.1 Evaluation Methodology

The Training, Test Sets and Cross-Validation

To evaluate the metric-based concept hierarchy construction framework, we use the framework to reconstruct concept hierarchy fragments from existing ontologies and compare the obtained concept hierarchy fragments with the ground truth. Particularly, we use concept hierarchy fragments extracted from WordNet and ODP as described in Chapter 3. 50 datasets from WordNet/*is-a*, 50 from WordNet/*part-of*, and 50 from ODP/*is-a*. Each concept hierarchy fragment contains concepts and relations between the concepts. We can also transfer the ontologies in to pairs of concepts.

With the extracted ontologies from WordNet and ODP, we create both training and test sets. The training data is in the same format of an extracted concept hierarchy: a set of concepts and a set of pairwise relations between the concepts. The test data only contain the concepts in the corresponding concept hierarchy and the relations need to be identified by an automatic concept hierarchy construction system.

We use leave-one-out cross validation to average the system performance across different training and test sets. For each 50 datasets from WordNet hypernyms, WordNet meronyms or ODP hypernyms, we pick 49 of them to generate the training data, and then test on the remaining dataset. We repeat the process for 50 times, with different training and test sets each time, and report the averaged system performance across all 50 runs.

Measurements

We use Fragment-Based Similarity (FBS) presented in Section 3.5 as one of the measurements to evaluate the system performance.

Moreover, both a constructed concept hierarchy and a gold standard concept hierarchy can be converted into a list of parent-child pairs if we enumerate all the pairs of parent and child nodes in the concept hierarchy. Note that even if a concept hierarchy is a "part-of" concept hierarchy, it still has parent-child nodes since the concept hierarchy is a tree. With a list of node pairs from a concept hierarchy, we can compare it with a list generated from the gold standard. We measure Precision, Recall, and F1-measure for the two lists.

Baseline Systems

We compare the metric-based concept hierarchy construction system with three state-of-the-art automatic concept hierarchy construction systems.

The first baseline system is a re-implementation of the Hearst system proposed by Marti Hearst with 6 hypernym patterns [Hea92]. We call this baseline system *HE*. In particular, this system uses a hand-crafted list of hypernym patterns, such as "*NPx, and/or other NPy*" and "*NPy including NPx*", as seeds and employs bootstrapping to discover new instances and patterns for *is-a* relation. The instances were successfully used to verify and augment the WordNet noun hierarchy as reported in [Hea92].

The second baseline system used in the evaluation is a re-implementation of the system proposed by Girju et al. (2003) [GBM03]. We call the system *GI*. The system is for the *part-of* relations. *Part-of* relation is a bit more complicated than *is-a* relation because there are many different kinds of part-whole structures. The system uses 3 most common meronym patterns [GBM03]. The patterns are either phrase-level patterns as in "high heel shoes", "girl's mouth", "eyes of a baby", and "door knob", or sentence-level patterns as in "the wheel is part of the car" and" the car contains four wheels".

The third baseline system is proposed by Snow et al. at Standford University [SJN06]. We call it *PR* since it is a probabilistic framework. This system also take an incremental approach to construct ontologies. However, in their work, a concept hierarchy grows based on maximization of conditional probability of relations given evidence; while in our work it grows based on optimization of concept hierarchy structures and modeling of concept abstractness and concept coherence. Moreover, our approach employs heterogeneous features from a wide range while they used only syntactic dependency. To have a fair comparison with *PR*, we extend their work and estimate the conditional probability of a relation given the evidence $P(R_{ij}|E_{ij})$ by using the same set of features as in our system.

5.5.2 Performance of Automatic Concept Hierarchy Construction

Our system is called *ME*, which is implemented based on the metric-based concept hierarchy construction framework described in this chapter.

Table 5.2 shows Precision, Recall, F1-measure, and FBS of the three baseline systems and our system for WordNet hypernyms (*is-a*), WordNet meronyms (*part-of*) and ODP hypernyms (*is-a*). The ground truth is generated from WordNet and ODP fragments. Bold font indicates the best performance in a column. Note that *HE* is not applicable to the *part-of* relation, and *GI* is not applicable to the *is-a* relation.

We observe that the proposed system *ME* consistently outperforms all baseline systems

Table 5.2: System performance (measured in Precision, Recall, F1-measure, and FBS).

WordNet/*is-a*				
System	*Precision*	*Recall*	*F1-measure*	*FBS*
HE	**0.85**	0.32	0.46	0.63
GI	n/a	n/a	n/a	n/a
PR	0.75	0.73	0.74	0.70
ME	0.82	**0.79**	**0.82**	**0.92**
ODP/*is-a*				
System	*Precision*	*Recall*	*F1-measure*	*FBS*
HE	0.31	0.29	0.30	0.60
GI	n/a	n/a	n/a	n/a
PR	0.60	**0.72**	0.65	0.72
ME	**0.64**	0.70	**0.67**	**0.83**
WordNet/*part-of*				
System	*Precision*	*Recall*	*F1-measure*	*FBS*
HE	n/a	n/a	n/a	n/a
GI	**0.75**	0.25	0.38	0.50
PR	0.68	0.52	0.59	0.71
ME	0.69	**0.55**	**0.61**	**0.81**

and produces the best F1-measures and FBS scores for all three tasks. Moreover, the F1-measures and the FBS scores are high, especially for WordNet is-a, which shows that our approach is very effective.

We also notice that systems using heterogeneous features (*ME* and *PR*) achieve higher F1-measure than systems only using patterns (*HE* and *GI*) with a significant absolute gain of >30%. There is also a significant absolutoin gain of 30%-40% for FBS. Generally speaking, pattern-based systems show higher precision and lower recall, while systems using heterogeneous features show lower precision but higher recall. However, when considering both precision and recall, using heterogeneous features is more effective than just using patterns. The results are statistically significant ($p < 0.005$, t-test).

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.

```
**ALGORITHM 5.5.3: Multi-Criteria Optimization w/o Abstractness.**
$W = \min_W \sum_{x=1} \sum_{y=1}^{|N(c_{training_x})|} \left( d_{xy} - \sqrt{\Phi(c_{training_x}, c_{training_y})^T W^{-1} \Phi(c_{training_x}, c_{training_y})} \right)^2$;
foreach $c_z \in C \setminus S$
    $S \leftarrow S \cup \{c_z\}$;
    if $W \succeq 0$
        $d(c_z, .) = \sqrt{\Phi(c_z, .)^T W^{-1} \Phi(c_z, )}$;
    $R \leftarrow R \cup \{\arg\min_{R_{(c_z,.)}} (\lambda u + (1 - \lambda) v)\}$;
Output $T(S, R)$
```

Figure 5.12: The algorithm for multi-criteria optimization w/o abstractness.

In Table 5.2, ME does not show a significantly better performance than PR. This is because we use the same of heterogenous features for both systems. For the original PR system presented in [SJN06], only syntactic dependency features are used and it would not achieve such a good performance. Even though, ME consistently outperform PR.

5.5.3 Impact of Concept Abstractness

This experiment studies the impact of modeling concept abstractness on system performance. Particularly, we test on two options: with or without optimizing the abstractness objective. With the abstractness objective being optimized, we just use Algorithm 5.3.5 and learn hierarchy metrics for concepts based on their abstraction levels. Without the abstractness objective being optimized, we make no distinction between concepts at different abstraction levels. Instead, we just learn from all concepts and relations in the training ontologies for a test concept hierarchy. We follow a modified version of Algorithm 5.3.5 when not optimizing the abstract objective. It is shown in Algorithm 5.5.3.

Table 5.3 presents the system performance of ME using the above two options. The runs are indicated as “w/o abstractness” or “w/ abstractness”. We test the two options for the WordNet *is-a*, ODP *is-a*, and WordNet *part-of* datasets.

The table shows that with the abstractness objective being optimized, the system gains a 6%-10% absolute improvement in F1-measure and a 7%-10% absolute improvement in FBS as compared to without the abstractness objective being optimized. The improvements is statistically significant ($p < 0.005$, t-test). It indicates that modelling concept abstractness is important to improve the overall performance of an concept hierarchy construction system.

Table 5.3: Impact of concept abstractness.

WordNet/*is-a*				
	Precision	*Recall*	*F1-measure*	*FBS*
w/o abstractness	0.78	0.69	0.73	0.85
w/ abstractness	**0.82**	**0.79**	**0.82**	**0.92**
ODP/*is-a*				
	Precision	*Recall*	*F1-measure*	*FBS*
w/o abstractness	0.59	0.61	0.60	0.73
w/ abstractness	**0.64**	**0.70**	**0.67**	**0.83**
WordNet/*part-of*				
	Precision	*Recall*	*F1-measure*	*FBS*
w/o abstractness	0.59	0.52	0.55	0.72
w/ abstractness	**0.69**	**0.55**	**0.61**	**0.81**

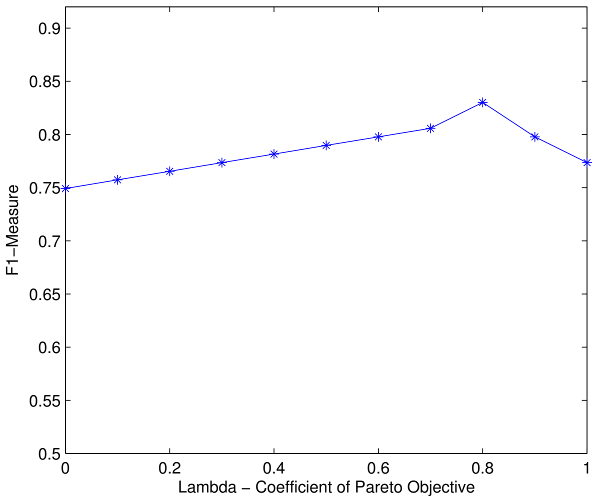
The proposed strategy of treating concrete concepts and abstract concepts differently is effective.

5.5.4 Impact of Concept Coherence

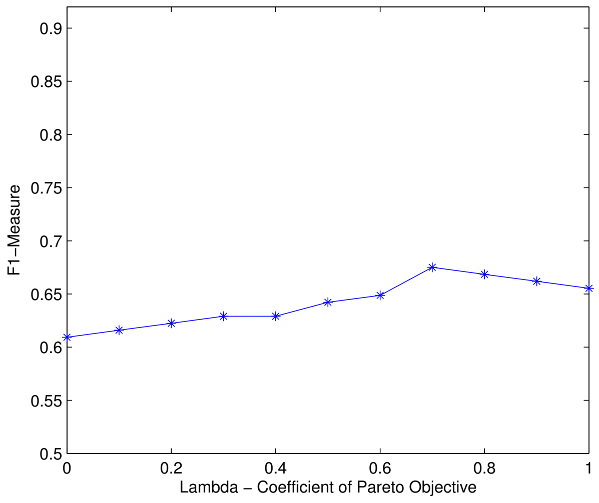
This experiment studies the impact of modeling concept coherence on system performance. Figure 5.13(a) and (b) show the changes of system performance on WordNet *is-a* and ODP *is-a* datasets when varying the coefficient λ in the Pareto objective as defined in Section 5.3.5. λ adjusts the contributions of minimum evolution objective and coherence objective. As $\lambda \to 0$, the system relies more on the coherence objective whereas as $\lambda \to 1$, the system relies more on the minimum evolution objective. λ is an indicator of the impact of concept coherence.

When $\lambda = 1$, the system purely relies on the minimum evolution objective. Figure 5.13 shows that as λ decreases, the contribution of concept coherence 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 coherence increases, but the system performance drops. The optimal λ values, 0.8 for WordNet and 0.7 for ODP, are used in experiments in Section 5.5.2, 5.5.3, 5.5.5, and 5.5.6.

It shows that a good combination of both objectives is important. Modeling concept coherence indeed improves the overall performance as compared to not modeling it at all (when



(a)WordNet/*is-a*



(b)ODP/*is-a*

Figure 5.13: Impact of Concept Coherence.

$\lambda = 1$). However, the major contribution still comes from minimum evolution objective.

5.5.5 Features vs. Relations

This experiment studies the impact of different types of features on different types of relations. We group the fifteen features in Section 5.4 into six sets: contextual, co-concurrence, patterns, syntactic dependency, word length difference and definition. Each feature set is turned on one by one for the following experiments.

Table 5.4 shows the F1-measure and FBS of using each set of features alone on automatic concept hierarchy construction for WordNet *is-a, sibling,* and *part-of* relations. Bold font indicates that a feature set makes a statistically significant contribution ($p < 0.005$, t-test) to automatic concept hierarchy construction for a particular type of relation.

The table 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 behaviors of contextual and syntactic features may be because that four out of five syntactic features (*Modifier*, *Subject*,

Table 5.4: F1-measure and FBS for Features vs. Relations: WordNet.

Feature	*is-a*		*sibling*		*part-of*		***Benefited Relations***
	F1	FBS	F1	FBS	F1	FBS	
Contextual	0.21	0.39	**0.42**	**0.82**	0.12	0.06	*Sibling*
Co-occurrence	**0.48**	**0.82**	**0.41**	**0.81**	**0.28**	**0.50**	*All*
Patterns	**0.46**	**0.39**	**0.41**	**0.80**	**0.30**	**0.62**	*All*
Syntactic	0.22	0.39	**0.36**	**0.76**	0.12	0.07	*Sibling*
Word Length	0.16	0.09	0.16	0.09	0.15	0.07	*All but limited*
Definition	0.12	0.07	0.18	0.11	0.10	0.05	*Sibling but limited*
All	***0.82***	***0.92***	***0.79***	***0.90***	***0.61***	***0.81***	*All*
Best Features	co-occurrence, patterns		contextual, co-occurrence, syntactic, patterns		co-occurrence, patterns		

Table 5.5: 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

Object, and *Verb* overlaps) are actually surrounding context for a concept.

The second last row of Table 5.4 shows the F1-measures and FBS scores for WordNet *is-a*, *sibling*, and *part-of* relations using all the features. We notice a statistically significant absolute gain in F1-measure (>10%) and FBS (10%-30%) by using all features than using any individual feature ($p < 0.001$, t-test). It indicates that combining heterogeneous features gives significant rise to the system performance than any single type of feature does.

5.5.6 Features vs. Abstractness

This section studies the impact of different sets of features on concepts at different abstraction levels. In the experiments, F1-measure and FBS are evaluated for concepts at each level of a concept hierarchy, not the whole concept hierarchy. Table 5.5 and Table 5.6 demonstrate

Table 5.6: 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

Table 5.7: FBS for Features vs. Abstractness: WordNet/*is-a*.

Feature	L_2	L_3	L_4	L_5	L_6
Contextual	0.33	0.62	**0.76**	**0.76**	**0.76**
Co-occurrence	**0.85**	**0.88**	**0.83**	**0.81**	**0.81**
Patterns	**0.85**	**0.81**	**0.81**	**0.77**	**0.81**
Syntactic	0.30	0.31	**0.77**	**0.76**	**0.81**
Word Length	0.09	0.09	0.09	0.09	0.09
Definition	0.07	0.07	0.07	0.07	0.07

F1-measure of using each set of features alone on each abstraction levels. Table 5.7 and Table 5.8 demonstrate FBS for the same set of experiments. Columns 2-6 in the table show the results for abstraction levels 2-6. The larger the indices are, the lower the levels. Higher levels contain abstract concepts, while lower levels contain concrete concepts. L_1 is ignored since it only contains a single concept, the root. Bold font indicates good performance in a column.

All tables show that abstract concepts and concrete concepts favor different sets of features. We find that contextual, co-occurrence, pattern, and syntactic features work well for concepts at L_4, L_5, and L_6, i.e., concrete concepts; co-occurrence works well for concepts at L_2 and L_3, i.e., abstract concepts. This difference indicates that concepts at different abstraction levels have different characteristics; it confirms the desirable *abstractness* property mentioned in Section 5.1.

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 for hypernyms, and hence the *is-a* relations in WordNet are recognized more effectively by using

Table 5.8: FBS for Features vs. Abstractness: ODP/*is-a*.

Feature	L_2	L_3	L_4	L_5	L_6
Contextual	**0.71**	**0.71**	**0.74**	**0.38**	**0.50**
Co-occurrence	**0.75**	**0.75**	**0.76**	**0.42**	**0.40**
Patterns	0.26	0.28	**0.69**	**0.40**	**0.39**
Syntactic	0.10	0.10	0.30	**0.39**	**0.39**
Word Length	0.09	0.09	0.08	0.08	0.08
Definition	0.07	0.07	0.07	0.07	0.07

lexico-syntactic patterns; while ODP contains more noise, and hence it favors features that require less rigidity, such as the contextual features generated from the Web.

5.5.7 Stability

ME is a general framework which integrates many different kinds of features for concept hierarchy construction. As we can expect, features originating from the pattern-based approaches (Section 2.1.1) should be able to produce stable relations and stable concept hierarchies since their decisions are merely boolean-valued. On the contrary, many other features used in ME, such as co-occurrence, contextual, syntactic features, are data-driven and originate from clustering-based approaches (Section 2.1.2). A known problem with concept hierarchies generated by clustering-based approaches which make use of the data-driven features is that when there are slight changes to a given document set, the concept hierarchy built by these approaches show dramatic changes. Because the data-driven features contribute to ME and ME is essentially still an incremental *clustering* algorithm, we want to know whether concept hierarchies generated by ME also suffer from this instability problem.

To evaluate the stability of concept hierarchies constructed by the metric-based concept hierarchy construction algorithm, we compare the similarity between concept hierarchies with slight document set changes. The metric used to measure hierarchy similarity is FBS.

We perform the stability test on the WordNet *is-a*, ODP *is-a*, as well as the Web datasets. The document sets for WordNet and ODP are part of the Wikipedia auxiliary datasets that are used to generate Wikipedia Global/Local Context KL-Divergence features (Section 5.4.1). We choose Wikipedia documents for their cleanness. Basically, we use the top 3 Wikipedia documents returned by Indri for a concept in a WordNet or ODP dataset to

Table 5.9: Stability of Concept Hierarchies (Measured by FBS).

	mean	stdv.	max	min
Web	0.90	0.03	0.94	0.85
WordNet	0.88	0.06	0.96	0.71
ODP	0.89	0.04	0.95	0.78

form the document set for this dataset. In general, a document set for WordNet or ODP contains about 120 documents (about 40 concepts *times* 3 documents per concept). We also perform this stability test on the Web datasets which are described in Section 3.3.2. Each Web document set is a crawled Web search result set which contains about 100 documents for a Web search topic as described in Table 3.2. In summary, we examine 50 WordNet, 50 ODP, and 5 Web (105 in total) document sets for the stability test. Each document set is about 100 to 120 documents.

We create slightly different document sets for each document set by sampling with replacements and calculate similarities between concept hierarchies built for these slightly changed document sets as the stability score. The procedure is described as follows:

1. For each document set, sample K ($K = 85$) out of N (N is the document set size, around 100 to 120) documents. Repeat the sampling with replacement for 5 times to get 5 slightly different sampled document sets.

2. Extract around 40 concepts based on the techniques presented in Chapter 4 and construct a concept hierarchy by ME for each sample document set.

3. Calculate the similarity between two concept hierarchies generated for two sample document sets. For the 5 samples created for an original document set, we can obtain similarity scores between 10 (5 choose 2) pairs of hierarchies.

4. The stability score for a document set is the average of these 10 hierarchy similarity scores.

We report the mean, standard deviation, max and min values for the stability scores for the 50 WordNet *is-a*, 50 ODP *is-a*, and 5 Web datasets in Table 5.9. Across different types of document sets, the stability scores between slightly changed sampled document sets are in the high-end - around 0.9 - in terms of FBS. This shows that concept hierarchies generated by ME are stable for slighted alternated document sets.

We believe that there are two main reasons lying behind the stability. Our approach exhaustively extracts concepts from a sampled document set and then filters and unifies some of them. Different from most cluster labeling techniques, where concepts are selected from a small partition of documents which clustered together, our approach discovers concepts within the entire document set. This makes the concept selection more stable because more documents are taken into account and less dramatic changes to the document partitions (only 1 partition in our case) happen. Therefore, the concepts shown in our hierarchies are quite stable when the document set changes slightly.

The second reason might be an even more important one. It is related to ME's ability to incorporate a wide range of features. The benefit of ME is that it not only has data-driven features such as contextual features and co-occurrence, but also has semantically meaningful features such as lexico-syntactic patterns, word definition, and modifier overlap. The semantically meaningful features originate from the pattern-based approaches. They are able to better capture the semantics among concepts as well as somehow decide the relations among concepts as if in a rule-based system. This kind of decision is deterministic. They help the concept hierarchies produced by ME remain stable even when a document set changes.

5.6 Summary

This chapter presents a novel metric-based concept hierarchy construction framework which incrementally clusters concepts and transforms the task of concept hierarchy construction into a multi-criteria optimization based on minimization of concept hierarchy structures, modeling of concept abstractness, and modelling of concept coherence. 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 concept hierarchy construction 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 is able to generate stable concept hierarchies as well as has the potential to learn more complex ontologies than previous approaches.

Automatic concept hierarchy construction produces ontologies without human intervention. The automatically-built ontologies maintain a fixed set of concepts and relations, which are not able to adapt to one's personal preference. In the next chapter, we present how to put human into the loop and construct personalized ontologies.

Chapter 6

Human-Guided Concept Hierarchy Construction

Personal concept hierarchy construction serves two goals: the first is to organize information into concept hierarchies, the second is to customize the concept hierarchies in the way that a user wants. Many Web search and text analysis situations require that a concept hierarchy not only well-represents the content and the scope of the topics in a document collection, but also suits an individual's specific needs. Concept hierarchies constructed automatically are not able to adapt to an individual user's needs nor to special use cases because no opinion or guidance from the user is considered. To support user-specific and task-specific concept hierarchies, it is necessary to study how to take into account the user's personal preferences in organizing the information.

This chapter presents Human-Guided Concept Hierarchy Construction. Specifically, this chapter studies how to incorporate user preferences in the concept hierarchy construction process, how to allow the machine learning algorithm to learn from the user, and how to produce a concept hierarchy according to the user's guidance. The framework is expected to produce concept hierarchies that reflect personal preferences as a consequence of learning from manual guidance.

The most challenging part of incorporating manual guidance in the machine learning process is how to translate it into a format that the machine can easily understand and incorporate into its learning models. In particular, we convert a concept hierarchy from a tree to matrices of neighboring nodes and represent the differences in matrices before and after human edits as manual guidance. We then train the learning framework to adjust to it and make predictions for unorganized concepts.

The metric-based concept hierarchy construction framework (Chapter 5) learns initial distance functions from group/community opinions in existing concept hierarchies which are constructed by a group of experts. In this chapter, human-guided concept hierarchy construction uses user-feedback to adapt and refine these distance functions to better match user preferences and task requirements.

This chapter consists of the following sections. We present an overview and a high-level algorithm for human-guided concept hierarchy construction in Section 6.1. We then present in Section 6.2 how to collect, represent, and translate manual guidance into a format that a machine learning algorithm can easily follow and understand. Afterwards we describe in Section 6.3 how the machine learns a distance function and makes prediction to organize the concepts according to the manual guidance that a user provides. The evaluation of this framework is detailed in Section 6.4.

6.1 The Human-Guided Concept Hierarchy Construction Framework

Human-guided concept hierarchy construction is an interactive process. Given a set of concepts, the machine first organizes the concepts and presents an initial concept hierarchy. In this dissertation research, the initial concept hierarchy are constructed by the automatic metric-based concept hierarchy construction framework presented in Chapter 5. Starting from the initial concept hierarchy, a user can teach the machine by providing manual guidance to it. The machine learns from the manual guidance and adjusts the distance learning function and modifies the concept hierarchy accordingly. The teaching and the learning alternate until the user is satisfied with the concept hierarchy. This concept hierarchy contains both the user's inputs and the machine's adjusted organization for the concepts.

Figure 6.1 shows an example of typical cycles of human-computer interactions in this framework. In this example, the cycle starts when the machine presents an initial concept hierarchy that consists of three concept groups: *person*, *hunter* and *habitat*. The user makes a

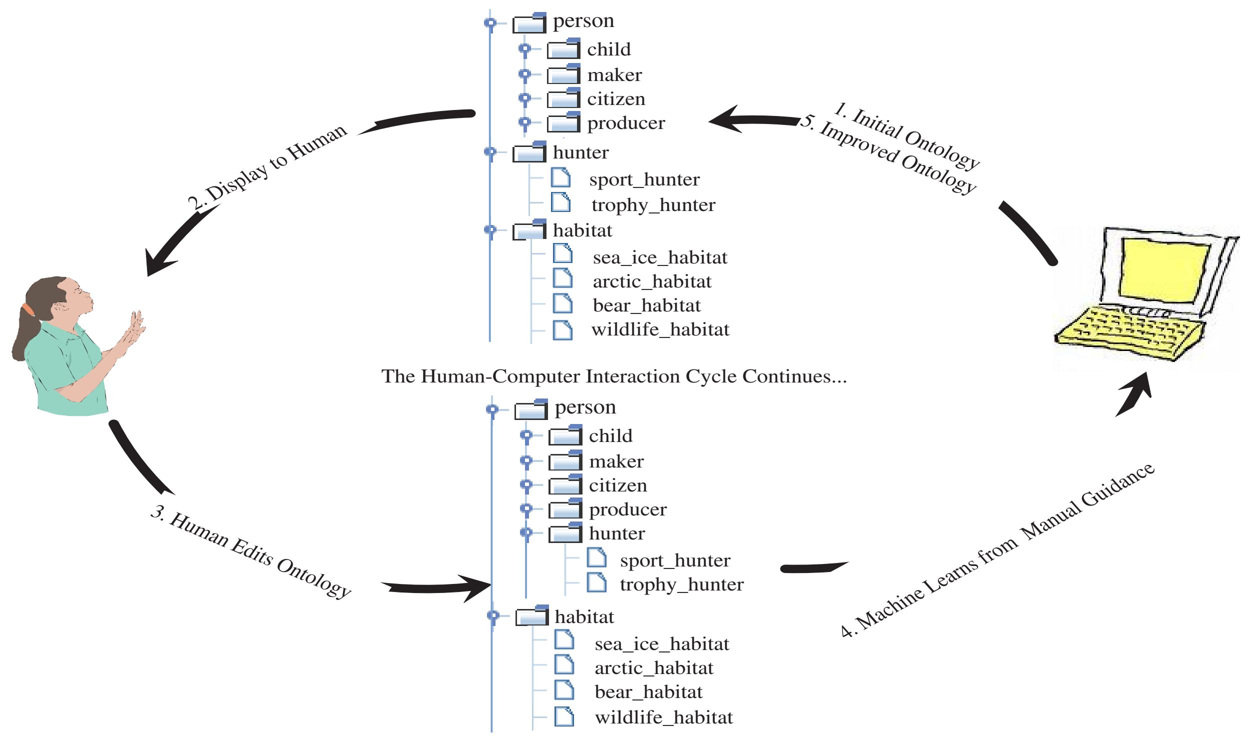


Figure 6.1: The human-computer interaction cycle.

modification to the concept hierarchy 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, makes modifications, and shows an improved concept hierarchy to the human. The human-computer interaction cycle continues until the user is satisfied with the concept hierarchy.

Algorithm 6.1 provides the pseudo codes for human-guided concept hierarchy construction. Line 1 of Algorithm 6.1 indicates the creation of an initial concept hierarchy by the machine (Chapter 5). Line 2 shows the initiation of three variables, U, G, and M, which are indexed by the iteration number i. U is the set of concepts which have not been modified by the user so far. U is initiated to be the entire set of concepts C, which can be acquired by the techniques presented in Chapter 4. G is the set of concepts and relations which have been modified by the user; it is initiated as empty. M is the manual guidance, the modifications that made by the user in the current iteration; it is initiated as empty, too. In summary, U keeps track of the concepts that the user has not visited or modified yet, G keeps track of

Algorithm 6.1: Human-Guided Concept Hierarchy Construction

```
1. CreateInitialConceptHierarchy();
2. $U^{(0)}$={Unmodified Concepts}=$C$, $G^{(0)}$={Modified concepts}=$\emptyset$, $M^{(0)} = \emptyset$,$i = 0$;
3. while (not Satisfied) or $U^{(i)} \neq \emptyset$
4.     $M^{(i)}$=CollectManualGuidance($G^{(i)}$,$U^{(i)}$);
5.     $F^{(i)}$=LearnDistanceMetricFunction($M^{(i)}$);
6.     $D^{(i)}$=PredictDistanceScores($F^{(i)}$,$U^{(i)}$);
7.     ($G^{(i+1)}$, $U^{(i+1)}$) = UpdateConceptHierarchy($D^{(i)}$,$U^{(i)}$,$G^{(i)}$);
8.     $i = i + 1$;
9. end
10. output $G^{(i)}$ as the concept hierarchy.
```

the concepts that the user has visited or modified, and M is calculated by the algorithm as a machine understandable format of guidance which reports the modifications made by the user in the current iteration i.

Line 3 to Line 9 in Algorithm 1 correspond to the human-computer interaction cycle. In particular, 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 unmodified concepts U, and obtains distance scores D for them. Line 7 shows that the machine organizes the unmodified concepts and updates the concept hierarchy with more modified concepts. Line 3 states the stopping criteria.

Finally, in Line 10, the algorithm outputs the latest modified set of concepts (with relations) G as the concept hierarchy, in which all concepts are organized based on their relations.

6.2 Collecting Manual Guidance

Human-guided concept hierarchy construction needs to obtain manual guidance from a user through human-computer interaction. It is challenging to collect manual guidance from the user without degrading her experience of organizing concept hierarchy. Collecting manual guidance with little interruption to the user's activity is one of the major concerns in designing a user interface. We use OntoCop (Chapter 3) to collect manual guidance. It is a tool with a user interface allowing a user to freely move concepts around and organize them with ease. In Section 6.2.1, we describe more functions of OntoCop for collecting manual guidance as well as interacting with users.

It is also challenging to represent manual guidance in a format that a learning algorithm can easily understand and incorporate it into the learning framework. We discuss in more details in Section 6.2.2 on how to represent concept hierarchy as matrix, and in Section 6.2.3 on how matrix representation can be used to collect manual guidance.

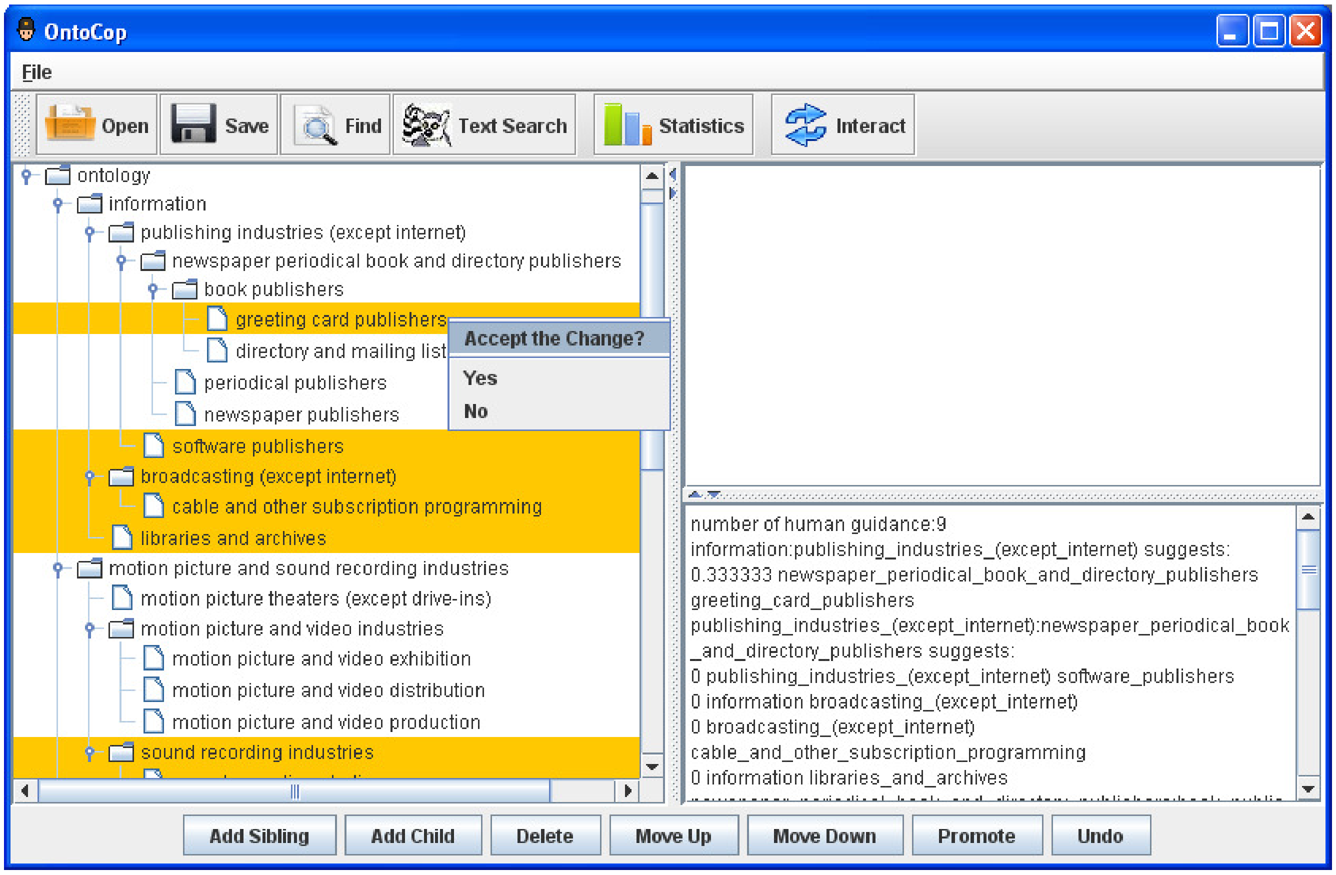
6.2.1 Interaction through OntoCop

Chapter 3 introduces the basic editing functions of OntoCop. In this section we focus on its functions that handle human-computer interactions.

Figure 3.1 shows a screen capture of the user interface of OntoCop. The last button on OntoCop's upper toolbar is the "interact" button. If the user clicks this "interact" button, the current edition of the concept hierarchy is submitted to the system, then the system learns from the user's most recent edits, updates the learning models, makes suggestions in an improved concept hierarchy and shows it to the user. The improved concept hierarchy is then displayed to the user with highlights to the system-suggested concepts within a few seconds.

Figure 6.2 illustrates the screen capture of a machine-updated concept hierarchy with highlights. The user can evaluate the suggestions by right-clicking any highlighted concept. If she thinks that a suggestion is valid, she can accept the suggestion by selecting the "yes" option from a drop-down menu which asks "(Do you want to) Accept the change?". If the user is not satisfied with a suggestion made by the system, she can reject it by selecting the "no" option from the drop-down menu. She can then provide more guidance for the next iteration if necessary.

The user is not required to make all modifications that she thinks necessary at once; she can make only a few modifications at each human-computer interaction cycle. When the user finishes a few modifications to the concept hierarchy, she triggers the system to take over by clicking the "interact" button on the toolbar. The system then learns from the user and suggests an improved concept hierarchy to her. The human-computer interaction cycle continues until the human is satisfied with the concept hierarchy. Note that the human modifications create different versions of a concept hierarchy. Each version is treated as an


OntoCop
File
Open
Save
Find
Text Search
Statistics
Interact
ontology
information
publishing industries (except internet)
newspaper periodical book and directory publishers
book publishers
greeting card publishers
directory and mailing list
periodical publishers
newspaper publishers
software publishers
broadcasting (except internet)
cable and other subscription programming
libraries and archives
motion picture and sound recording industries
motion picture theaters (except drive-ins)
motion picture and video industries
motion picture and video exhibition
motion picture and video distribution
motion picture and video production
sound recording industries
Accept the Change?
Yes
No
Add Sibling
Add Child
Delete
Move Up
Move Down
Promote
Undo

```
number of human guidance:9
information:publishing_industries_(except_internet) suggests:
0.333333 newspaper_periodical_book_and_directory_publishers
greeting_card_publishers
publishing_industries_(except_internet):newspaper_periodical_book
_and_directory_publishers suggests:
0 publishing_industries_(except_internet) software_publishers
0 information broadcasting_(except_internet)
0 broadcasting_(except_internet)
cable_and_other_subscription_programming
0 information libraries_and_archives
```

Figure 6.2: System suggestions in OntoCop.

independent concept hierarchy.

6.2.2 Matrix Representation of Concept hierarchies

OntoCop uses a tree structure to store and manage a concept hierarchy. However, trees are not straightforward for a machine learning algorithm to manipulate. In order to capture the changes between each version of the manual editions, the learning algorithm needs both the training and the test data to be in a format which is easy to handle. Matrix representation can be easily understood and manipulated by many machine learning algorithms. We therefore convert concept hierarchies from trees to matrices and use a matrix representation for all the intermediate editions in the concept hierarchy construction process.

In this dissertation research, we use an *hierarchy matrix* to represent a concept hierarchy.

Formally, a concept hierarchy with n concepts can be represented by a $n \times n$ hierarchy matrix. Each row and each column of the hierarchy matrix corresponds to a concept in the concept hierarchy. The entries in the matrix indicate whether (or how confident) a relation r is true for the concepts. Specifically, the $(i, j)^{th}$ entry of an *hierarchy matrix* indicates the confidence in $r(c_i, c_j)$. The value of the $(i, j)^{th}$ entry v_{ij} is defined as:

$$v_{ij} = \begin{cases} 1, & \text{if } i = j, \\ 1, & \text{if the relation } r \text{ is true between } c_i \text{ and } c_j,\ i \neq j, \\ 0, & \text{if the relation } r \text{ is not true between } c_i \text{ and } c_j,\ i \neq j. \end{cases} \tag{6.1}$$

where r is a type of relation between the concepts. The positive entries in the hierarchy matrix indicate that the relation between two concepts is true, and the zero entries indicate that the relation between two concepts is false. If no confidence level is used, we simply employ boolean values as entries in a concept hierarchy matrix.

In general there can be any relation between concepts. Depends on the types of relations of interest, a concept hierarchy can be represented as *is-a hierarchy matrix*, *sibling hierarchy matrix*, *part-of hierarchy matrix* or other types of concept hierarchy matrices. Different relations may result in different hierarchy matrixes for the same dataset.

6.2.3 Defining the Manual Guidance

With the matrix representation, we can compare the changes in concept hierarchy matrices. They are essential to understand manual guidance. Particularly, manual guidance can be collected by comparing a concept hierarchy before and after human modifications, which indicate a user's preferences about how to construct a personal concept hierarchy. The procedure for extracting manual guidance from a relation-specific matrix is described below.

We represent the organization of concepts before a user's modifications as a *before matrix*; likewise, the new organization of concepts after her modifications is represented as a *after matrix*. Given these two matrixes, *manual guidance* is a submatrix in *after matrix* that shows the differences between *before matrix* and *after matrix*.

Figure 6.3 illustrates a concept hierarchy which contains five concepts - {*person, leader, president, prime minister, Obama*}. The concepts are in the political domain and the relation type is *sibling*. The *before matrix A* for the concept hierarchy in Figure 6.3 can be represented

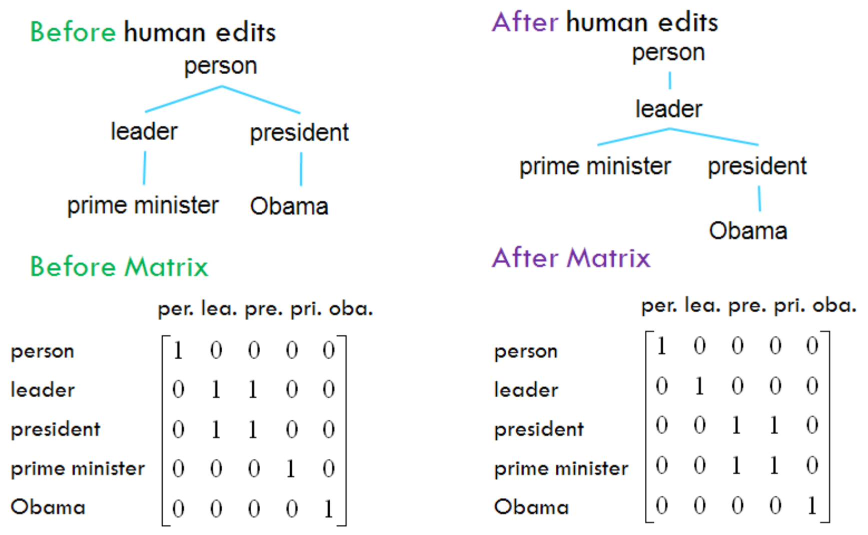


Figure 6.3: An example concept hierarchy before and after human modifications (Concept set unchanged; relation type = *sibling*).

as:

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

Although a user can make multiple changes to the concept hierarchy during one iteration, the user makes only one change in this example. She moves the node "president" to be under "leader"; 'president"'s child node "Obama" also moves together with it. After human modifications, the example concept hierarchy can be represented as an *after matrix* B:

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

We the compare the before matrix A and the after matrix B to derive the manual guidance M. The manual guidance is not simply the matrix difference between the before matrix and the after hierarchy matrix. It is part of the after matrix because it is the after matrix that indicates where the user wants the concept hierarchy to develop. We define manual guidance M as a submatrix which consists of some entries of the *after matrix* B; at these entries, there exist differences 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.

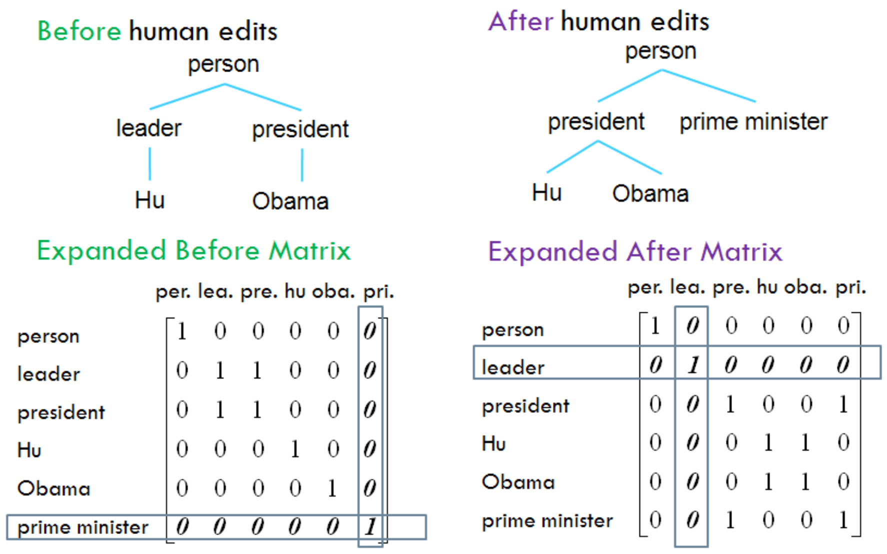
For the example in Figure 6.3, the difference between B and A is:

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

The positive entries in the difference matrix indicate the user's preference on how to group the corresponding concepts together; the negative entries indicate her preference of keeping the corresponding concepts apart. In this example, 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 differences between A and B. The sign is not important since we only care about the differences. Hence, manual guidance M is:

$$M = B[2,3,4; 2,3,4] = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} = \begin{array}{cccc} & \textit{leader} & \textit{president} & PM \\ \textit{leader} & 1 & 0 & 0 \\ \textit{president} & 0 & 1 & 1 \\ PM & 0 & 1 & 1 \end{array}.$$

This simple example illustrates the case when the set of concepts remain unchanged before and after human modifications. Many human modifications produce unchanged concept set,



$$\begin{array}{c|cccccc} & per. & lea. & pre. & hu & oba. & pri. \\ \hline person & 1 & 0 & 0 & 0 & 0 & 0 \\ leader & 0 & 1 & 1 & 0 & 0 & 0 \\ president & 0 & 1 & 1 & 0 & 0 & 0 \\ Hu & 0 & 0 & 0 & 1 & 0 & 0 \\ Obama & 0 & 0 & 0 & 0 & 1 & 0 \\ prime\ minister & 0 & 0 & 0 & 0 & 0 & 1 \end{array}$$

$$\begin{array}{c|cccccc} & per. & lea. & pre. & hu & oba. & pri. \\ \hline person & 1 & 0 & 0 & 0 & 0 & 0 \\ leader & 0 & 1 & 0 & 0 & 0 & 0 \\ president & 0 & 0 & 1 & 0 & 0 & 1 \\ Hu & 0 & 0 & 0 & 1 & 1 & 0 \\ Obama & 0 & 0 & 0 & 1 & 1 & 0 \\ prime\ minister & 0 & 0 & 1 & 0 & 0 & 1 \end{array}$$

Figure 6.4: An Example Concept Hierarchy Before and After Human Modifications (Concept set changes; relation type = *sibling*).

for example, dragging and dropping, moving up and moving down, and promoting concepts. However, oftentimes the user adds, deletes or renames concepts, and the concept set changes. When the concept set changes, the above definition of manual guidance M needs a slight alteration.

Figure 6.4 shows another example concept hierarchy whose concept set changes. The original concept set before the human modification is {*person, leader, president, Hu, Obama*}. The concept hierarchy's *before matrix* A is:

$$A = \begin{array}{cccccc} & person & leader & president & Hu & Obama \\ person & 1 & 0 & 0 & 0 & 0 \\ leader & 0 & 1 & 1 & 0 & 0 \\ president & 0 & 1 & 1 & 0 & 0 \\ Hu & 0 & 0 & 0 & 1 & 0 \\ Obama & 0 & 0 & 0 & 0 & 1 \end{array}.$$

The user modifies the concept hierarchy at several places. In particular, *leader* is deleted, *Hu* is moved to be under *president*, and *prime minister* is inserted as a new concept into this concept hierarchy. Therefore the concept set changes to {*person, president, Hu, Obama,* *prime minister*}. The *after matrix* B is:

$$B = \begin{array}{c|ccccc} & person & president & Hu & Obama & PM \\ person & 1 & 0 & 0 & 0 & 0 \\ president & 0 & 1 & 0 & 0 & 1 \\ Hu & 0 & 0 & 1 & 1 & 0 \\ Obama & 0 & 0 & 1 & 1 & 0 \\ PM & 0 & 1 & 0 & 0 & 1 \end{array}.$$

Since the concept sets before and after the human modifications change, we cannot simply use matrix subtraction to get the difference between the before and after matrices. Suppose the concept set in the concept hierarchy before the modifications is C_A, and the concept set after modifications is C_B, we define an expanded set of concepts C_E as the union of C_A and C_B:

$$C_E = C_A \bigcup C_B.$$

We then define an *expanded before matrix* A' and an *expanded after matrix* B' 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 6.4, the *expanded before matrix* A' is:

$$A' = \begin{array}{c|cccccc} & person & leader & president & Hu & Obama & PM \\ person & 1 & 0 & 0 & 0 & 0 & 0 \\ leader & 0 & 1 & 1 & 0 & 0 & 0 \\ president & 0 & 1 & 1 & 0 & 0 & 0 \\ Hu & 0 & 0 & 0 & 1 & 0 & 0 \\ Obama & 0 & 0 & 0 & 0 & 1 & 0 \\ PM & 0 & 0 & 0 & 0 & 0 & 1 \end{array}.$$

Note that the expanded concept set C_E is {*person, leader, president, Hu, Obama, prime minister*}. The 6^{th} row and the 6^{th} column are newly expanded. They correspond to the concept *prime minister*, which is newly added to the concept hierarchy.

The *expanded after matrix* B' is:

$$B' = \begin{array}{lcccccc} & person & leader & president & Hu & Obama & PM \\ person & 1 & 0 & 0 & 0 & 0 & 0 \\ leader & 0 & 1 & 0 & 0 & 0 & 0 \\ president & 0 & 0 & 1 & 0 & 0 & 1 \\ Hu & 0 & 0 & 0 & 1 & 1 & 0 \\ Obama & 0 & 0 & 0 & 1 & 1 & 0 \\ PM & 0 & 0 & 1 & 0 & 0 & 1 \end{array}.$$

Note that the 2nd row and the 2nd column are newly expanded. They correspond to concept *leader*, which is deleted from the concept hierarchy.

For concept hierarchies with concept changes, we define the manual guidance M as a submatrix which consists of some entries of the *after matrix* B; at these entries, there exist differences 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 6.4, the difference between B' and A' is:

$$B' - A' = \begin{array}{lrrrrrr} & person & leader & president & Hu & Obama & PM \\ person & 0 & 0 & 0 & 0 & 0 & 0 \\ leader & 0 & 0 & -1 & 0 & 0 & 0 \\ president & 0 & -1 & 0 & 0 & 0 & 1 \\ Hu & 0 & 0 & 0 & 0 & 1 & 0 \\ Obama & 0 & 0 & 0 & 1 & 0 & 0 \\ PM & 0 & 0 & 1 & 0 & 0 & 0 \end{array}.$$

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

$$M = B[2,3,4,5;2,3,4,5] = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} = \begin{array}{lcccc} & \textit{president} & \textit{Hu} & \textit{Obama} & \textit{PM} \\ \textit{president} & 1 & 0 & 0 & 1 \\ \textit{Hu} & 0 & 1 & 1 & 0 \\ \textit{Obama} & 0 & 1 & 1 & 0 \\ \textit{PM} & 1 & 0 & 0 & 1 \end{array} .$$

6.3 Predicting the Relations

Manual guidance indicates a user's preference of how to organize the concepts into a personalized concept hierarchy. It provides guidance for the interactive machine learning algorithm to further organize other concepts to agree with the user. Basically, we use it as training data in the human-guided concept hierarchy construction framework. In Section 6.3.1, we present how to learn a new distance function during each interaction based on the manual guidance. We then describe how to predict the distances between the unmodified concepts in Section 6.3.2 and how to organize those unmodified concepts based on the learned distances in Section 6.3.3.

6.3.1 Learning the Distance Function

The human-guided concept hierarchy construction framework employs a supervised distance learning algorithm to learn user preferences from manual guidance. The algorithm trains and directs the learning models towards the user preferences and then predict new groupings for the unmodified concepts. This section presents the supervised distance learning algorithm.

Manual Guidance as the Training Data

Section 6.2 presented how to collect manual guidance M from the human. Based on M, we can create training data for a supervised distance learning algorithm. In particular, we

	president	*Hu*	*Obama*	*PM*
president	0	1	1	0
Hu	1	0	0	1
Obama	1	0	0	1
PM	0	1	1	0

Figure 6.5: Training distance matrix.

transform the manual guidance into a distance matrix and the distance matrix is used as the training data.

Recall that manual guidance M contains the concepts modified by the user and the relations the user determines for these concepts. The entries in M indicate whether a relation r is true for two concepts at a particular row and a particular column. If the relation is true, the two concepts should be connected together according to r, and their distance is 0. In M, larger values indicate that two concepts are close to each other (their relation is true) and smaller values indicate that they are further apart. In a distance matrix, larger values mean that two concepts are further apart and smaller values mean that they are close to each other and should be grouped together. Therefore, the distance matrix is the opposite of the manual guidance. We transform manual guidance M to a distance matrix D as follows:

$$D = 1 - M \tag{6.2}$$

The relation type presented in the distance matrix is determined by the relation type presented by the manual guidance. For example, if the relation r is *is-a*, in a training distance matrix, the parent-child pairs are indicated as 0, and other nodes are indicates as 1. If the relation r is *sibling*, within-cluster distances are defined as 0 and between-cluster distances are defined as 1.

Figure 6.5 shows the training distance matrix derived from the example in Section 6.2.3. This training distance matrix elaborates the distance between *president*, *Hu*, *Obama* and *PM (prime minister)*. It is used as the training data for a supervised distance learning algorithm.

A Supervised Distance Learning Algorithm

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. We use the same distance learning method as proposed in Section 5.3.1. The difference between the distance learning in this chapter and in Chapter 5 is that in this chapter we directly use manual guidance to derive the training data while in Chapter 5 we use existing concept hierarchies such as WordNet and ODP as the training data.

According to Algorithm 6.1, human-guided concept hierarchy construction has three major variables, the unmodified concepts U, the modified concepts G, and the manual guidance M. At each iteration of the human-computer interaction, the user groups the concepts in G by dragging and dropping, or by using other editing functions. The machine learns a distance function from concepts in M and 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 unmodified concepts, modified concepts and manual guidance, respectively. The unmodified concepts are those concepts which are not connected as any other concepts' parent, child or sibling in the previous i iterations. The training data consists of the set of concepts in $\mathbf{G}^{(i)}$ and their corresponding pair-wise distance matrix $\mathbf{D}^{(i)}$.

Given concepts $C = \{c_1, c_2, \ldots, c_n\}$, we organizes these concepts and outputs a concept hierarchy $T(C', R)$. C' is the final set of concepts, which closely relate to C but do not necessarily equal to C since our framework allows changing the concept set by adding or deleting concepts, or changing names of concepts. However, for simplicity, we just use C to denote concepts in this section.

Similar to Equation 5.21, we apply minimization of squared error and constrain the weight matrix $W^{(i)}$ for the i^{th} iteration to be positive semi-definite, the optimization function for the parameter estimation is formulated as:

$$W^{(i)} = \min_W \sum_{x=1}^{|\mathbf{G}^{(i)}|} \sum_{y=1}^{|\mathbf{G}^{(i)}|} \left(d_{xy} - \sqrt{\Phi(c_x^{(i)}, c_y^{(i)})^T W^{-1} \Phi(c_x^{(i)}, c_y^{(i)})} \right)^2 \tag{6.3}$$

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

where d_{xy} is the abbreviation of $d(c_x^{(i)}, c_y^{(i)})$, $\Phi(c_x^{(i)}, c_y^{(i)})$ represents a set of pairwise underlying feature functions, $W^{(i)}$ is the i^{th} weight matrix for the i^{th} human-computer interaction, which weighs the underlying feature functions at the i^{th} iteration.

After $W^{(i)}$ is learned from the manual guidance, we use it to predict the distance scores for the unmodified concepts and further group them accordingly. The initial training from WordNet and ODP is smoothed with this new training from the user.

Feature Representation

Both modified and unmodified concepts use the same feature representation. Each pair of concepts are represented by a feature vector $\overrightarrow{x}$, which contains numerical scores of features such as patterns, co-occurrence, definition, contextual, and syntactic parse features (Section 5.4).

6.3.2 Predicting Distance Scores for Unmodified Concepts

To organize the concepts to agree with user preferences, the system learns from manual guidance and predicts the labels for the unmodified pairs. The learning model predicts whether two concepts $c_x \in U$ and $c_y \in U$ have the relation r true between them. For example, if r is "sibling", it decides whether c_x and c_y belong to the same concept group. If r is "is-a", it decides whether c_x is the parent node of c_y. Note that a sibling relation is symmetric while a parent-child relation is asymmetric. The learning model uses all concept pairs in $G^{(i)}$ to estimate a weight matrix $W^{(i)}$ based on Equation 6.3.

Given the learned parameter matrix $W^{(i)}$ in the i^{th} iteration, we can generate distance scores for any pair of unmodified concepts in U^i. By calculating the distance for each concept pair, we obtain the entries in a new distance matrix $\hat{\mathbf{D}}^{(i+1)}$ for the $i+1^{th}$ iteration. Note that this distance matrix should also result in a consistent clustering, which is guaranteed by the positive semi-definiteness of the parameter matrix $W^{(i)}$. The entry values for $\hat{\mathbf{D}}^{(i+1)}$ is defined as:

$$\hat{d}_{lm}^{(i+1)} = \sqrt{\Phi(c_l^{(i+1)}, c_m^{(i+1)})^T W^{(i)-1} \Phi(c_l^{(i+1)}, c_m^{(i+1)})} \tag{6.4}$$

where $d_{lm}^{(i+1)}$ is the abbreviation of $d(c_l^{(i+1)}, c_m^{(i+1)})$, and $(c_l^{(i+1)}, c_m^{(i+1)})$ is an unmodified concept pair from $U^{(i)}$.

The learned distance matrix $\hat{\mathbf{y}}^{(i+1)}$ contains the distance scores for concepts in $U^{(i)}$.

6.3.3 Organizing Concepts into Updated Concept Hierarchies

Based on the predicted distance scores, we group the unmodified concepts in a concept hierarchy. When a pair-wise distance score is small (<0.5), we consider the relation between the concept pair is true.

How to organize the concepts whose relation is true, is decided again by the relation type in the distance matrix. If r is "sibling", c_l and c_m are put into the same concept group. If r is "is-a", c_m is put under c_l as one of c_l's children.

The newly modified concepts are added into $G^{(i)}$ and form a new set of modified concepts $G^{(i+1)}$. 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 concept hierarchy to the human and waits for the next round of manual guidance.

6.4 Evaluation

We conduct a user study to evaluate effectiveness and efficiency of human-guided concept hierarchy construction. We aim to evaluate how effective is the interactive approach as compared to a manual approach to construct concept hierarchies. We asked the users to evaluate the suggestions made by OntoCop at each interaction cycle as well as tell us how well the system learns from their human edits. As additional evidence of how well the concept hierarchies are built by the users using OntoCop, we compare constructed concept hierarchies with the reference concept hierarchies built manually by experts. We also evaluate the efficiency of OntoCop by evaluating how much time and editing effort can be saved by using the system. The evaluation is based on the final concept hierarchies created by the users, their on-the-fly judgements to the system, and an after-task questionnaire.

In this section, we describe the tasks, procedure, datasets, and experimental results of this user study.

6.4.1 Tasks

The user study involved 24 participants who are mainly undergraduate and graduate students from Carnegie Mellon University and the University of Pittsburgh. They were required to use

Participant's Role	**Task**
Rule maker	Exploring important issues raised in a public comment set (5 tasks).
Concept hierarchy constructor	Organizing concepts in a particular NAICS domain (10 tasks).
Web user	Planning a trip to DC.
Groom-to-be/Bride-to-be	Finding a good wedding videographer in the Pittsburgh area.
New parent to cook for your son's 1st month party	Finding out how to make a cake.
Poor graduate student	Finding useful information for buying a used car in the Pittsburgh area.
Parent of a toddler	Finding a good kindergarten in the Pittsburgh area.

Table 6.1: Participants' roles and tasks.

OntoCop to construct concept hierarchies for browsing document collections with real-life tasks in mind.

When constructing a concept hierarchy for a dataset, the participants were asked to bear in mind a particular task. Specifically, they were assigned tasks to organize concepts in a document set[1]. Example tasks and roles include "planning a trip to DC as if you were an ordinary Web user", "find a good wedding videographer as if you were a groom-to-be/bride-to-be", "organizing information in the domain of financial businesses", and "exploring issues mentioned in a public comment set as if you were a rule maker". The complete roles and tasks are listed in Table 6.1.

For each task, the participants started from a flat list of concepts and used OntoCop to organize them into concept hierarchies. During each task, the participants either construct a concept hierarchy manually or interacted with OntoCop to construct the concept hierarchy interactively.

6.4.2 Procedure

The user study was conducted in sessions. Each session is two hour long. The users were first introduced to OntoCop for about 10 minutes so that they could get familiar with its

[1]In the follow-up questionnaire, several participants mentioned that they would like to use the software to write survey papers, which suggests that it might be the task that they thought they were performing

functions. This training was then followed by another exercise task which lasted about 15 minutes. Afterwards, users started the real tasks and worked on the tasks for 90 minutes. The tasks include both manual and interactive runs. Once the real tasks are done, users had 5 minute to answer a questionnaire regarding their experience.

To separately evaluate human-guided concept hierarchy construction and metric-based concept hierarchy construction, the participants started from a flat list of concepts and used OntoCop to organize them into concept hierarchies. Each participant was assigned to construct concept hierarchies manually for half of the datasets, and interactively for the other half; so that each participant had a chance to use both methods. We adopt a Latin Square design and the order of construction methods were randomized to avoid order effects.

For the manual runs, a participant had access to most functions of OntoCop, such as dragging and dropping, and renaming a concept. However, she did not have access to the "Interact" function. For the interactive runs, the participant have access to all functions including "Interact". A typical interactive run is as follows: a participant did a few edits in a human-computer-interaction, and then clicked the "Interact" button. The system then learned from this human edits and promoted some suggestions. These suggestions were highlighted for the participant and she could then choose to either 'accept' or 'reject' the suggestions. This choice is an on-the-fly evaluation to this iteration of system's learning and prediction. After that, the participant could either stop the process if she was satisfied with the concept hierarchy or continued to update the concept hierarchy by making a few more changes, and 'interact' with the system in the next iteration. The entire construction of a concept hierarchy is finished when the participant felt satisfied with the concept hierarchy or reached a 20-minute time limit for each task.

Note that the completion of a task was mainly decided by the participants, who stopped when feeling satisfied with a construction. The 20-minutes limit was very generous. We believe this is necessary for the participants to freely organize concepts in a way that they personally liked, with no time pressure.

After completion of each task, the participants answered a few questions to qualitatively evaluate the system's performance and their own user experience. Example questions include *"How do you evaluate the difficulty to organize concept hierarchies for each dataset?"*, *"How confident are you about the quality of your edits to the concept hierarchies"*, and *"How well*

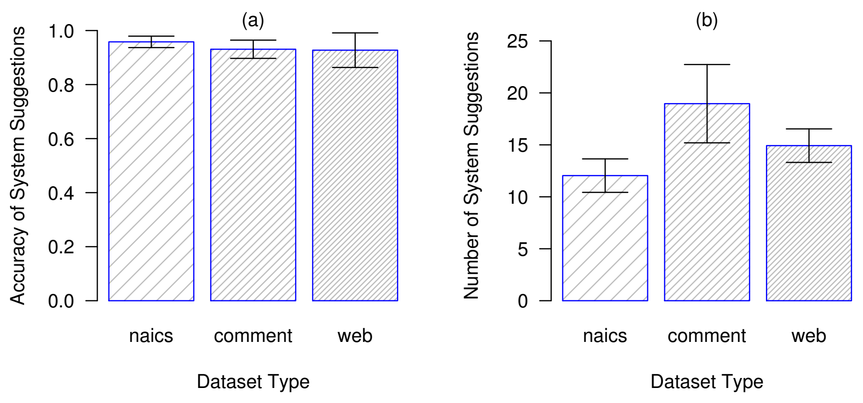


Figure 6.6: Mean accuracy of system suggestions and mean number of system suggestions (Interactive runs only).

did the system appear to learn your method of organizing the concept hierarchies? (Only for datasets that you organized using the Interact function)".

Similarly, after completion of all tasks, the participants were asked to answer a short after-study questionnaire, including qualitative questions such as *"Do you like constructing concept hierarchies with interaction with the software? (yes, no, maybe)"*, *"Do you think interacting with the software helps you construct a better hierarchy? (yes, no, maybe)"*, and free form comments.

We include the complete questionnaires in Appendix A.

6.4.3 Datasets

We evaluate our system for a variety types of datasets. We used 10 NAICS datasets, 5 public comment datasets, and 5 Web datasets. Each dataset is one task for the participants to construct a concept hierarchy about it. Each dataset contains 40 concepts in order to fit into one screen. The details of the datasets are in Chapter 3.

6.4.4 Accuracy of System Suggestions

To evaluate the human-guided concept hierarchy construction framework presented in this chapter, we measure the accuracy of system suggestions. The participants constructed concept hierarchies for half of the tasks manually, and the other half interactively. During every human-computer interaction cycle, the system made suggestions based on a participant's edits and the suggestions were evaluated by the participant according to her own standard. She could judge a suggestion by selecting an option "yes" or "no" from the "Accept the change?" menu (Figure 6.2). This on-the-fly evaluation directly reflects how well the system learns from human edits. A high accuracy indicates that the system learns well from user edits and the user accepts many of the suggestions. In particular, the accuracy of system suggestions is calculated as:

$$\text{Accuracy} = \frac{1}{r}\sum_{i=1}^{r}\frac{\text{number of accepted suggestions in } i^{th} \text{ cycle}}{\text{number of suggestions in } i^{th} \text{ cycle}} \tag{6.5}$$

where r is the total number of human-computer interaction cycles when constructing a concept hierarchy.

Figure 6.6 (a) shows the mean accuracy and its 95% confidence interval of the system suggestions, broken down by dataset types. The accuracy is at least 0.92 for all datasets, and 0.94 on average, which is high. Note that the participants did not select "yes" to everything. This high accuracy demonstrates that the system successfully learns from a participant and makes highly-accurate predictions on how the participant would organize the concepts. It shows that human-guided concept hierarchy construction is effective.

Figure 6.6 (b) illustrates the average number and its 95% confidence interval of suggestions made by the system when constructing a concept hierarchy. The average number of suggestions across all datasets is 15.3. It indicates that about 38% of the relations in a finalized concept hierarchy were suggested by the system, and among them at least 92% were accepted by the participants as correct suggestions.

We notice that the system made different number of suggestions to different types of datasets. For public comment datasets, the average number of system suggestions is 18; for NAICS datasets, 10; and for the Web datasets, 15. In general the NAICS datasets receive less suggestions from the system than the public comment and the Web datasets. The reason

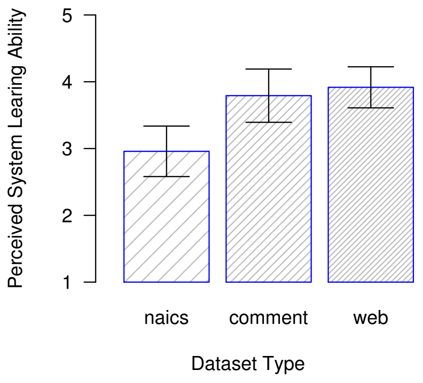


Figure 6.7: Perceived system learning ability (Interactive runs only).

may be related to dataset difficulty. We will discuss it in Section 7.3.2.

6.4.5 Perceived System Learning Ability

The participants worked on twenty tasks. After completion of each interactive task, a participant was asked immediately to rate how well the system learned from her edits in order to produce a concept hierarchy. The question was *"How well did the system appear to learn your method of organizing the concept hierarchies? (Only for datasets that you organized using the Interact function)"*. A rating in 5-point scale, ranging from "very good"(5), "good"(4), "fair"(3), "bad"(2), to "trash"(1), was used to rate perceived system learning ability.

Figure 6.7 shows the mean and 95% confidence interval for the perceived learning ability. The average system learning ability perceived by the participants is 3.61. If breaking it down by different types of datasets, the NAICS datasets have a mean perceived learning ability of 2.95, which is statistically significant lower than that for the Web (Mean=3.92) and the public comments datasets (Mean=3.79). One-way ANOVA test shows that there is a statistically significant difference for different dataset type on the perceived learning ability ($p < .015$).

From the after session questionnaire, we found that participants thought that NAICS datasets were more difficult and they were not familiar with this domain. It is interesting that when a dataset is less familiar or more difficult for the users, the system was perceived

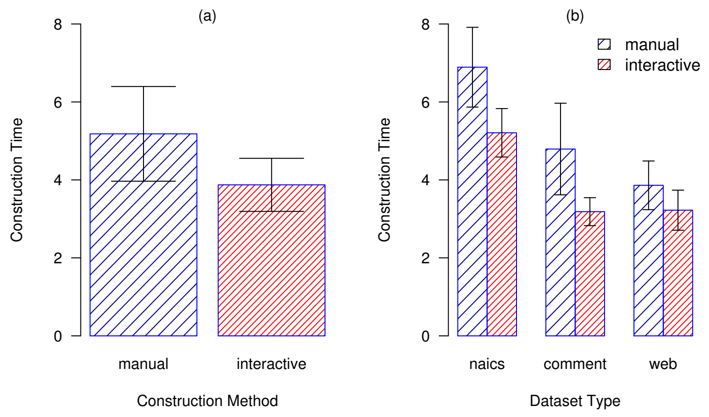


Figure 6.8: Mean hierarchy construction time (in minutes; both manual and interactive runs).

to perform badly too. The statistically significant difference between NAICS and the other two types of datasets may suggest that when people are not familiar with the tasks, they provide less promising edits, the system learns from the lower quality training data, and in the end the participants perceive the output as poor system learning ability. We study dataset difficulty more in Section 7.3.2.

6.4.6 Efficiency

Both the accuracy of system suggestions directly judged by the participants and the perceived system learning ability show that the human-guided concept hierarchy construction framework is effective in learning from manual guidance. We also concern with the efficiency of using the interactive system since it uses computational power and tries to save manual effort. In this experiment, we examine the construction time and the number of edits a user needs in both the interactive runs and the manual runs.

Construction Time

Figure 6.8 (a) shows the average time (and its 95% confidence interval) used to construct a concept hierarchy by different construction methods. For the interactive runs, the average construction time that the participants used is 3.87 minutes. For the manual runs, the

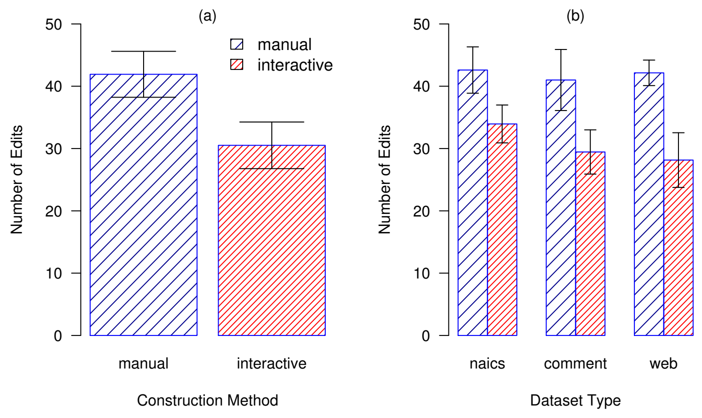


Figure 6.9: Mean number of edits (For both manual and interactive runs).

average construction time is 5.18 minutes. We perform statistical significance tests to analyze the construction time. The results show that the interactive method used statistically significantly less time (1-min or 20% less per dataset on average) than the manual construction method ($p < .001$ on a one-way ANOVA test). It indicates that human-guided concept hierarchy construction can greatly reduce the time needed in concept hierarchy construction.

Figure 6.8 (b) shows the average time (and its 95% confidence interval) used to construct a concept hierarchy for different dataset types (including both manual and interactive runs). For the NAICS datasets, the average construction time is 6.05 minutes, the public comment datasets 3.54 minutes, and the Web datasets 3.54 minutes. It is not surprising that participants spent statistically significant ($p < .001$ in a one-way ANOVA test) more time to finish constructing concept hierarchies with NAICS datasets than the other two datasets since NAICS datasets are more 'difficult'. On average, participants spent about 2 minutes (or 20-30%) more on an NAICS dataset than other datasets.

Number of Edits

The types of edits that a participant made to construct the concept hierarchies include dragging and dropping a node, adding a node, deleting a node, renaming a node, promoting a node, and undoing an editing action. The number of edits a participant used to construct a concept hierarchy is an indicator of her manual effort for the construction. We study how the human-guided concept hierarchy construction framework can save users' editing effort.

Figure 6.9 (a) shows the average number (and its 95% confidence interval) of edits used to construct a concept hierarchy by different construction methods. For the interactive runs, the average number of edits that the participants used is 31. For the manual runs, the average number of edits is 42. The interactive method results in statistically significantly fewer human edits than the manual method ($p < .001$ in a one-way ANOVA test). Given that the size of each concept hierarchy is around 40 nodes, the interactive runs save about 25% human edits by suggesting groupings and organization for concepts.

Figure 6.9 (b) shows the average number (and its 95% confidence interval) of edits used to construct a concept hierarchy for different dataset types. For the NAICS datasets, the average number of edits is 38, for public comment datasets is 35, and for the Web datasets is 35 too. The number of edits for different types of datasets are not statistically significantly from each other. Dataset type does not play a role in difference in number of edits.

6.4.7 Comparing to Reference Concept Hierarchies

Concept hierarchy construction is a personalized task. Evaluating how good is a personalized concept hierarchy is subjective and usually can only be judged by the person who constructs it. However, we also notice that people share some commonality in organizing concepts and may want to share their personal concept hierarchies with each other. Therefore, how much a concept hierarchy is similar to or different from other concept hierarchies gives us more ideas about whether a user constructs the concept hierarchy successfully enough to represent a reasonable organization of the concepts. We hence use concept hierarchies created by experts or popular concept hierarchies agreed by many participants as reference concept hierarchies and compare the concept hierarchies constructed by the participants against the reference. This measurement is not to measure whether a concept hierarchy satisfies a user's need, which we measured by system suggestion accuracy and perceived learning ability. This measurement is to compare how different a concept hierarchy is from a reference concept hierarchy which is conducted by experts or agreed by many people.

We use Fragment-Based Similarity (FBS), the concept hierarchy similarity measure proposed in Chapter 3, to measure the similarity between a concept hierarchy created by a participant, either manually or interactively, and a reference concept hierarchy. For the 10

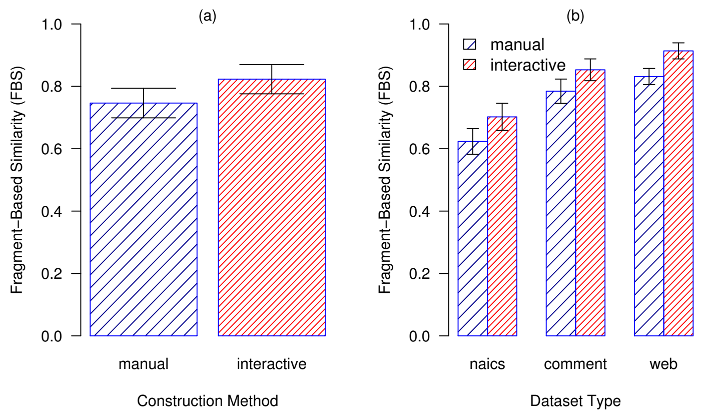


Figure 6.10: Fragment-based similarity (FBS) against reference concept hierarchies (For both manual and interactive runs).

NAICS datasets, we used the 2007 NAICS codes as the reference concept hierarchies. For the Web and the public comment datasets, we use the most popular concept hierarchy, i.e., the concept hierarchy whose mean FBS to other concept hierarchies is the highest among the participants' concept hierarchies pool.

In Figure 6.10 (a), we plot the mean similarity (and its 95% confidence interval) between a concept hierarchy and a reference concept hierarchy using different construction methods. The interactive runs produces an averaged similarity of 0.82, while the manual runs produces an averaged similarity of 0.74. The difference between the interactively constructed concept hierarchies and the manually constructed concept hierarchies are statistically significant ($p < 0.001$ in a one-way ANOVA test). This result shows that when the participants interact with the human-guided concept hierarchy construction system, they produce concept hierarchies more similar to reference concept hierarchies than when they work manually without help from the system. In general, we observe greater consistency among participants when they used the interactive system.

This result implies that the manual runs allow a user to follow her own idea more freely while the interactive runs somehow lead the user towards a concept hierarchy which is more agreeable among different people. It may attribute to the fact that the interactive runs suggested new organizations of concepts to a user, and some suggestions were accepted by the user. The system not only learns from the user, but also the user updates her views with what the system suggests. It is a two-way learning through human-computer interactions. The interactive system encourages people to be more consistent by suggesting choices that are consistent with their previous choices. In addition, this machine-teaching effect may also happen during the tool training where all participants were given the same training data to be familiar with OntoCop.

In Figure 6.10 (b), we plot the mean similarity (and its 95% confidence interval) between a concept hierarchy and a reference concept hierarchy for different dataset types. The NAICS datasets give an averaged similarity of 0.66, while the public comment datasets give 0.82 and the Web datasets give 0.87. The difference between the NAICS concept hierarchies and the other two types of datasets are statistically significant ($p < 0.001$ in a one-way ANOVA test).

This result implies that dataset type plays a significant role in the resulting concept hierarchies. In particular, the concept hierarchies created for NAICS are less similar to the reference concept hierarchies, as compared to the concept hierarchies created for the Web and the public comment datasets.

6.5 Summary

This chapter describes the human-guided concept hierarchy construction framework for constructing concept hierarchies interactively through human-computer interaction. By incorporating personal preferences as manual guidance, the proposed supervised distance learning algorithm and concept hierarchy construction framework is able to predict effectively and organize the concepts into concept hierarchies.

In human-guided concept hierarchy construction, 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 concept hierarchy, the machine learns from 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 a concept hierarchy to be based on the personal preferences from individuals.

Human-guided concept hierarchy construction has been tested on several different types of datasets. The evaluation on a variety of datasets give us a better idea of how well the framework works under different situations. The framework has successfully demonstrated its ability to deal with all these dataset types. In general, it is effective in terms of being highly accurate to make suggestions to the users and being perceived as learning well from the users. It also greatly saves the users' effort in terms of time and number of edits.

Chapter 7

Study of User Behaviors

Concept hierarchy construction incorporates personalization into concept hierarchy construction. It aims to construct concept hierarchies that satisfy user-specific or task-specific needs. In Chapter 6, we describe a user study that evaluates effectiveness and efficiency of the human-guided concept hierarchy learning framework. In this chapter, we continue to describe the user study from the users' perspective. Particularly, we study the user behaviors and how human, system, and dataset work together to product differences in the concept hierarchies being constructed.

We start the chapter with a list of research questions that we want to get answer in Section 7.1. We then identify the possible influencing factors in concept hierarchy construction through an exploratory analysis in Section 7.2. We set up experiments and perform statistical significance tests to evaluate whether these factors cause difference in how to organize information. Sections 7.3, 7.4, and 7.5 detail the experimental design and the results.

7.1 Research Questions

When people create concept hierarchies for a task, the subjectivity in how to organize the information determines that everyone probably creates a different concept hierarchy for the same task. Although it is possible for different people to create the entire concept hierarchy in common, the chance of this coincidence is slim. In this chapter, we are interested in the influencing factors of producing differences in concept hierarchies.

The possible influencing factors in creating different concept hierarchies may come from three sources. The first source is the construction methods, including manual and interactive methods. The second source is the datasets. The datasets belong to different types. In this user study, we use three types of datasets: NAICS, public comments, and the Web datasets. Datasets are different by nature and it is not surprising to see concept hierarchies created for different datasets or different dataset types are different. However, for datasets that belong to the same dataset type, the concept hierarchies created for them may show some commonalities within this type and show differences between the types. The third source of the influencing factors are the participants.

There are twenty-four participants in this user study. They are from various majors in Carnegie Mellon University and University of Pittsburgh. The age of the participants ranged from 19 to 33 years old (Mean=24.3, SD=2.38). Eleven participants were females (46%). All participants had basic computer skills and experience using software such as Microsoft Windows Explorer at least twice a week. All participants had completed at least two years of college, and were either native speaker (79%) or had high proficiency (21%) in English. Of the 24 participants, 12 were randomly selected to re-perform the tasks again three weeks later. Table 7.1 summarizes the statistics of the 24 participants in this user study.

The participants's different organizing or editing habits could be caused by their demographics, errors/inconsistencies due to sloppiness, or other personal preferences which are not captured by demographics. In our records, the difference between the participants range from their gender, major, and language proficiency. Some of these are possible influencing factors to generate the differences in the concept hierarchies.

With the possible influencing factors, we examine whether they are real influencing factors on concept hierarchies construction and on user behaviors by conducting statistically significance tests. We examine these factors on various aspects of concept hierarchy construction, including construction time, number of edits, quality of edits, and feature use.

We also study how consistent people are when they construct concept hierarchies. That is, to find out if the variation among users is really due to personal preferences instead of random variations.

In a nutshell, we evaluate various aspects of how people organize information and construct concept hierarchies through finding answers to the following research questions:

Institutes	
Carnegie Mellon University	16
University of Pittsburgh	8
Gender	
Male	13
Female	11
Majors	
Arts	3
Business	1
Chemical Engineering	2
Cognitive Science	2
Computer Science	8
Math	1
Public Policy	3
Psychology	4
English Proficiency	
Native speaker	19
Non-native speaker (with high proficiency)	5

Table 7.1: Statistics of participants.

Q1: What are the potential influencing factors (from the aforementioned three sources) for the differences of the concept hierarchies?

Q2: Do these factors make statistically significant differences between concept hierarchies and in user behaviors?

Q3: Whether people are self-consistent?

7.2 Influencing Factors

As stated in Section 7.1, the possible influencing factors of the differences among the concept hierarchies come from three sources: construction methods, datasets, and participants. Construction method and dataset type are obvious possible influencing factors. From the participant point of view, we need to find out the possible influencing *participant* factors in creating different concept hierarchies.

We take an exploratory approach to discover the participant factors. In particular, we quantitatively compare the similarity among the concept hierarchies and cluster the participants into groups based on how similar their concept hierarchies are. We then study the common characteristics inside each user group. These group-wise characteristics could be the factors that we are looking for.

We employ the following steps to form the user clusters.

1. *Compute a user similarity matrix.*

   - There are 20 datasets and 24 participants. For each dataset D_i, participant j creates an concept hierarchy T_{ij}. Among the 24 concept hierarchies created for D_i, we calculate the pair-wise Fragment-Based Similarity (FBS) scores for two concept hierarchies T_{ij} and T_{ik}, where $i = 1, ..., 20$, $j = 1, ..., 24$, and $k = 1, ..., 24$.
   - If $FBS(T_{ij},T_{ik}) > \sigma$, we consider that two concept hierarchies T_{ij} and T_{ik} are similar, and participant j and participant k create similar concept hierarchies for dataset D_i. σ is empirically set to 0.5.
   - We count the number of similar datasets created by two participants and form a user similarity matrix F. F is a 24×24 matrix with each row and each column representing a participant. An entry (j, k) in F indicates the number of datasets for which that participants j and k create similar concept hierarchies. In another word, F records on how many datasets two participants agree.

2. *Cluster the participants.*

   - With the user similarity matrix F, we can group the participants into clusters based on how many datasets they agree with each other. We use the *hclust* function in R programming[1] to perform agglomerative hierarchical clustering over F. The agglomeration option used is *ward*, a minimum variance method aims at finding compact, spherical clusters [War63].
   - The hierarchical clustering algorithm groups the participants into user clusters, where each cluster represents a group of participants who tend to create similar concept hierarchies.

[1]http://www.r-project.org/.

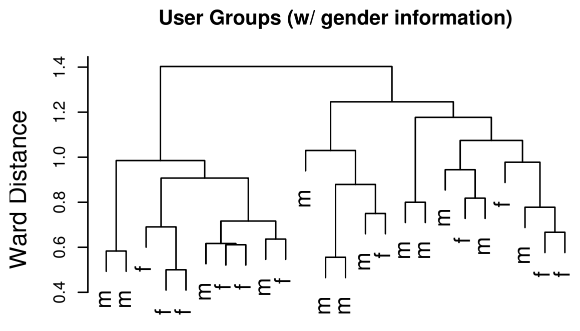
3. *Study the participants' characteristics within the user groups.*

   - For each user cluster, we mark the corresponding participants' user ids.
   - We also mark their gender, major, and language proficiency. Through observing what characteristics being shared within a user group, we infer possible influencing factors that make the users different between the clusters.

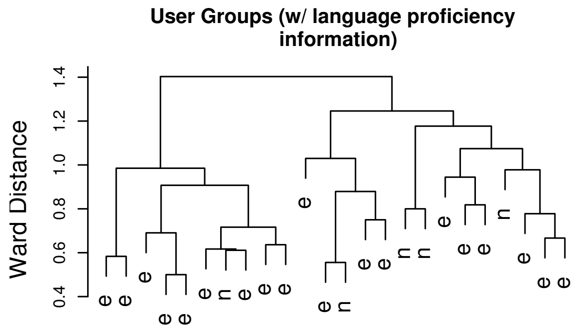
Figure 7.1 illustrates the results of the agglomerative hierarchical clustering. Overall, the participants are clustered into two big groups: one user group consists of 10 participants, and another group consists of 14. There are more finer groups being formed, however we do not have enough data to distinguish the subtle differences between the finer groups. Instead, we focus on the two big user groups and study their within-cluster commonality and between-cluster differences.

Figure 7.1 (a) tags the gender information for each participant in the hierarchical clustering results. The y-axis indicates values of the distance scores between the participants based on the Ward agglomeration method. Eleven females and thirteen males participated in the user study. The random female ratio is 0.46 and the random male ratio is 0.54. The 10-people group consists of six females out of ten, which gives a high female ratio of 0.6 within the group so that we consider this group as a “female” group. The 14-people group consists of nine males out of fourteen, which gives a high male ratio of 0.64 within the group so that we consider this group as a “male” group. It is interesting that the user groups are formed based on whether the users create similar concept hierarchies, but the clusters naturally show a slight gender difference. No matter whether there is a gender difference when constructing concept hierarchies, we think gender is an interesting possible influencing factor worth exploring in building concept hierarchies.

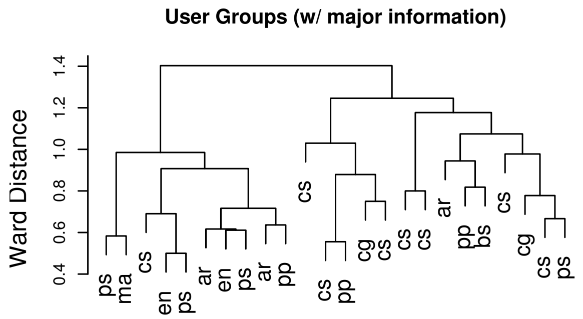
Figure 7.1 (b) tags the participants' language proficiency information. Nineteen native English speakers and five non-English speakers participated in the user study. The random native speaker ratio is 0.79, and the random non-native speaker ratio is 0.21. The non-English speakers all show high langauge proficiency as they reported and confirmed by us through conversations. The 10-people group consists of nine native speakers out of ten, which gives a high native speaker ratio of 0.9 so that we consider this group as a “native” group. However, the 14-people group consists of four non-native speakers out of fourteen,



(a) 'f' – female, 'm' – male.



(b) 'e' – native speaker, 'n' – non–native speaker.



(c) 'cs' – Computer science, 'ps' – Psychology, 'ma' – Ma
'en' – Engineering, 'ar' – Arts, 'pp' – Public policy,
'cg' – Cognitive science, 'bs' – Business.

Figure 7.1: Users form clusters based on how similar the concept hierarchies that they created are.

which gives a non-native ratio of 0.28 that is not significantly larger than the random ratio. Hence we can only consider this group as a "native" group too. The two user groups do not show difference in terms of language proficiency. We therefore ignore it as a possible influencing factor in concept hierarchy construction.

Figure 7.1 (c) tags the participants' majors in the hierarchical clustering results. The participants were from Computer Science, Mathematics, Engineering, Arts, Psychology, Cognitive Science, Business, and Public Policy. The random CS-major ratio is 0.33, and the non-CS major ratio is 0.67. We find that the 10-people group consists of nine non-CS majors out of ten, which gives a high non-CS ratio of 0.9 (significantly larger than 0.67) within the group so that we consider this group as a "non-CS" group. The 14-people group consists of seven CS majors out of fourteen, which gives a high CS ratio of 0.5 (significantly larger than 0.33) within the group so that we consider this group as a "CS" group. Similar to gender, it is interesting that CS majors and non-CS majors naturally fall into different user groups based on how they organize the concept hierarchies. At this point of time, we are not sure whether people majoring in CS indeed behave differently from people majoring in other non-CS majors. However, at least we can consider major as another possible influencing factor in creating different concept hierarchies.

In summary, we take an exploratory data-driven approach to investigate what the possible influencing factors are for the differences of the concept hierarchies being created by different people. We find that gender and major are two such possible influencing factors. Gender difference is not a new topic. It commonly appears in many tasks that involve humans. However, we would like to know whether it also appears in the task of concept hierarchy construction. Since CS-major apparently show differences from the other majors, we also would like to find out whether it is true that people majoring in CS think and organize information differently from people majoring in other majors do.

Recall that construction method and dataset type are two other possible influencing factors. Together with gender and major, we investigate whether these factors really impact on how people construct concept hierarchies by performing statistical significance tests.

7.3 Impact of the Factors on Concept hierarchies

Based on the possible influencing factors suggested in the previous section, we analyze how the participants organize information according to these factors. In this section, we compare the concept hierarchies by measuring their similarities as well as by studying the user behaviors. For example, we can compare how long it takes to create a concept hierarchy by different participants, what is one's use pattern of different types of editing functions, and what is one's use pattern of different types of feature functions.

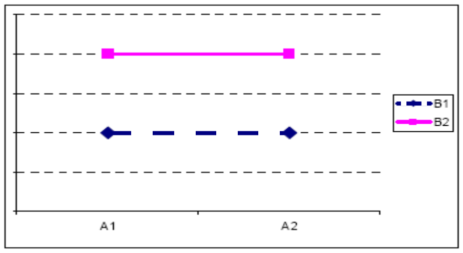
7.3.1 Experimental Design

We employ the repeated measure analysis of variance (ANOVA) tests [Lin74] to study the correlations and interactions among multiple influencing factors. In particular, we perform two-way ANOVA tests to evaluate the significance of the influencing factors, which we call *independent variables (IV)*, on various aspects of the concept hierarchies being constructed, which we call the *dependent variables(DV)*.

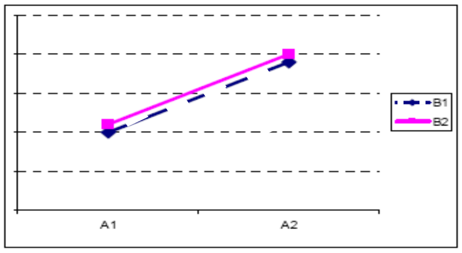
A two-way ANOVA test involve two independent variables and one dependent variable. Suppose the independent variables are A and B, the dependent variable is C. The test first evaluates A's individual effect on C and B's individual effect on C, independently from each other. If the test of A on C is statistically significant, i.e. the p-value is less than a significance cut-off threshold, we say A has a *main effect* on C. Similar for B on C. The main effect indicate a correlation between the independent variable and the dependent variable. The test then evaluates A and B's effects on C simultaneously. If the test of A and B on C is statistically significant, i.e. the p-value is less than a significance cut-off threshold, we say that A and B has an *interaction effect* on C.

Besides p-values, the significance of the tests can be observed through ANOVA interaction plots. They straightforwardly reveal the correlations and the interactions among the variables. Figure 7.2 illustrates a few interaction plots and their interpretations for several two-way ANOVA tests involving two IVs A and B. A has two values A_1 and A_2; B also has two values B_1 and B_2. The two color-coded lines represent B. The two end points at each line represent the mean value of A given B. As we observe, a separation of the two lines suggests a main effect of B (Figure 7.2 (a), (c), and (f)). A steep slope of the lines suggests a

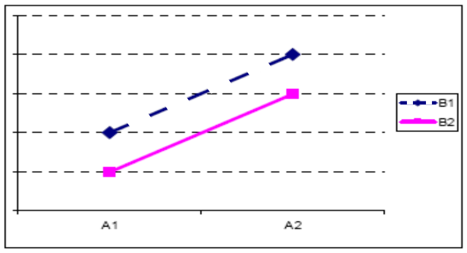
(a) No effect of A, main effect of B.



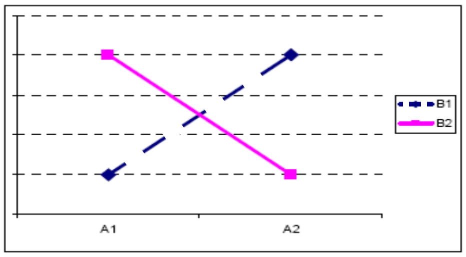
(b) Main effect of A, no effect of B.



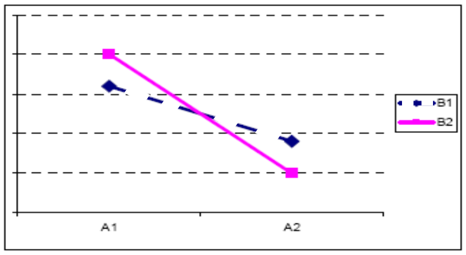
(c) Both main effects.



(d) No main effects, interaction effect.



(e) Main effect of A, no effect of B, slight interaction.



(f) Both main effects, interaction effect.

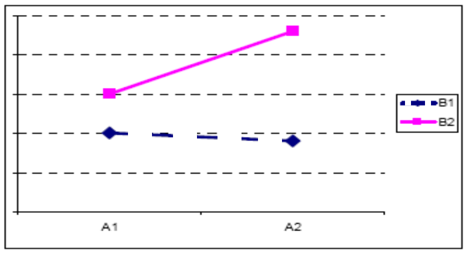


Figure 7.2: Interpretation of two-way ANOVA interaction plots. (Adapted from "Two-way ANOVA tests" by Agilent Technologies, 2005.)

main effect of A (Figure 7.2 (b), (c), (e), and (f)). The main effects could be cancelled out if the slopes are equal but with different signs (Figure 7.2 (d)). Moreover, a crossing of the two lines suggests an interaction effect of A and B (Figure 7.2 (d), (e), and (f)). Nonetheless, it is the p-value that decides the significance of the tests.

Specifically, in this user study, the independent variables include two between-subject independent variables: major (CS and non-CS[2]) and gender (male and female), and two within-subject independent variables: construction method (manually or interactively) and

[2]We do not study the major difference among every major, instead, since the user groups suggest that Computer Science major may show difference against other majors, we break down all the majors into CS and non-CS. Moreover, there is not enough data for each non-CS major to get reliable measurements.

Table 7.2: Summary of variables.

Variables	Description
Within-subject IV: Construction method	Manual; Interactive.
Within-subject IV: Dataset type	NAICS; Comments; Web.
Between-subject IV: Major	Computer science; Non-Computer science.
Between-subject IV: Gender	Male; Female.
DV:Dataset difficulty	Perceived dataset difficulty (1-5).
DV:Construction time	Time spent to construct a concept hierarchy (0-20 mins).
DV:Number of edits	The amount of human edits made to a concept hierarchy (0-∞).
DV:Perceived quality of edits	The quality of human edits perceived by a participant (1-5).
DV:Top features	The top weighted feature functions used by a participant (pattern, co-occurrence, context, syntactic, definition).
DV:Self-agreement	Measured in FBS (0-1).

dataset type (NAICS, Web, or public comments). These independent variables represent the potential influencing factors that could cause the differences in the constructed concept hierarchies and the user behaviors.

The dependent variables in the ANOVA tests represent different aspects of the concept hierarchies being created. We asked the participants to rate the task difficulty and the quality of their own edits for different datasets based on their perceptions. For the more objective measures, we recorded the time that a participant spent to construct a concept hierarchy, the number of edits that a participant made for a concept hierarchy, the features that a participant used, and for some participants their self-agreement in a follow-up study. We would like to know if there is a correlation or an interaction between these dependent variables and the independent variables. Table 7.2 list all the independent and dependent variables.

We perform multiple statistical significance tests according to the above independent and dependent variables. In total we perform three two-way ANOVA tests: 2 (construction method) $\times$ 3 (dataset type) ANOVA, 2 (construction method) $\times$ 2 (major) ANOVA, and 2 (construction method) $\times$ 2 (gender) ANOVA.

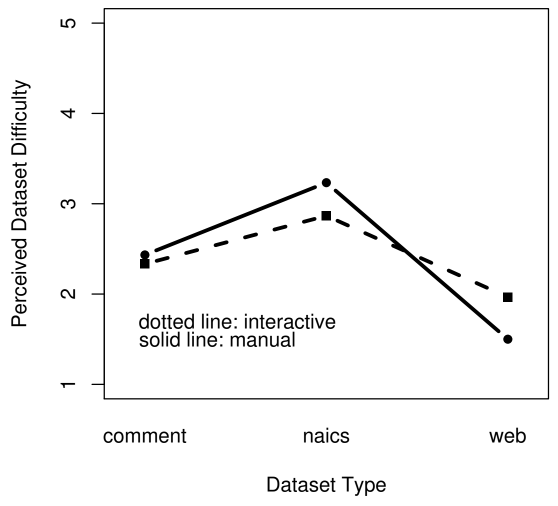


Figure 7.3: Interaction plot for mean perceived dataset difficulty.

These tests allow us to find out the significant correlations and interactions among different variables and help us understand which potential factors really cause the difference in the resulting concept hierarchies. We report the results in the following sections.

7.3.2 Dataset Difficulty

Whether a dataset is difficult for a participant is important, because it directly impacts on whether this participant can successfully create a reasonable concept hierarchy. One's personal opinion of whether a dataset is difficult or easy to deal with, depends on her own experience and how familiar she is with the domain. We provide the participants datasets of varying difficulty and collect the "perceived dataset difficulty", which is a 5-point scale score rated by every participant immediately after they completed the dataset. That participants thought that NAICS datasets were more difficult and they were not familiar with this domain. On the contrary, the public comments are closely related to their daily life. The comments are basically emails sent to the government agencies about environmental, transportation, and other living-condition related issues. Similarly, the Web datasets are Web queries that people sent to search engines. The topics such as "trip planning" are so familiar to the participants.

We perform three two-way ANOVA tests on the perceived dataset difficulty. The three tests are “construction method vs. dataset type”, “construction method vs. major”, and “construction method vs. gender”. We find that there is a statistically significant main effect for dataset type ($p < .001$) on perceived dataset difficulty. No main nor interaction effect was caused by major or gender.

Figure 7.3 shows the ANOVA interaction plot for the “construction method vs. dataset type” test. It consists of two lines, one representing the interactive method and another representing the manual method. The x-axis represents three different dataset types. The y-axis is the rating scores from 1 to 5. The three points on each line show the mean values of the perceived dataset difficulty for each dataset type. We observe a statistically significant main effect for dataset type ($p < .001$) on perceived dataset difficulty. Specifically, the difference comes from between the NAICS datasets vs. the other two dataset types. We find no statistical significant difference on perceived dataset difficulty between the Web dataset and the public comment datasets.

This result suggests that different types of datasets indicate different difficulty levels for the datasets. The NAICS datasets are more difficult for participants than the Web and the public comments datasets. One reason might be that both the Web and comments datasets contain familiar topics to the participants. For example, “make a cake”, “find a used car”, and “list polar bear as threatened species”. The topics in NAICS, for example “finance” and “administrative services”, are more distant from the participants’ daily life as they are nearly all students. The vocabulary in NAICS contains rare concepts such as “synthetic rubber manufacturing” and “non-depository credit intermediation”, some of which the participants complained in their free form comments as “never heard of”. Because the participants were not familiar with the concepts in NAICS, they felt that the NAICS datasets were hard and they could not well-organize the NAICS concepts into good concept hierarchies.

In Figure 7.3, the crossing of the two lines show that there is a slight interaction effect ($p < .01$) between construction method and dataset type on perceived dataset difficulty. In particular, we observe a raise of the perceived difficulty in the NAICS datasets and a drop of the perceived difficulty in the Web datasets in the interactive runs; we observe the opposite trends in the manual runs.

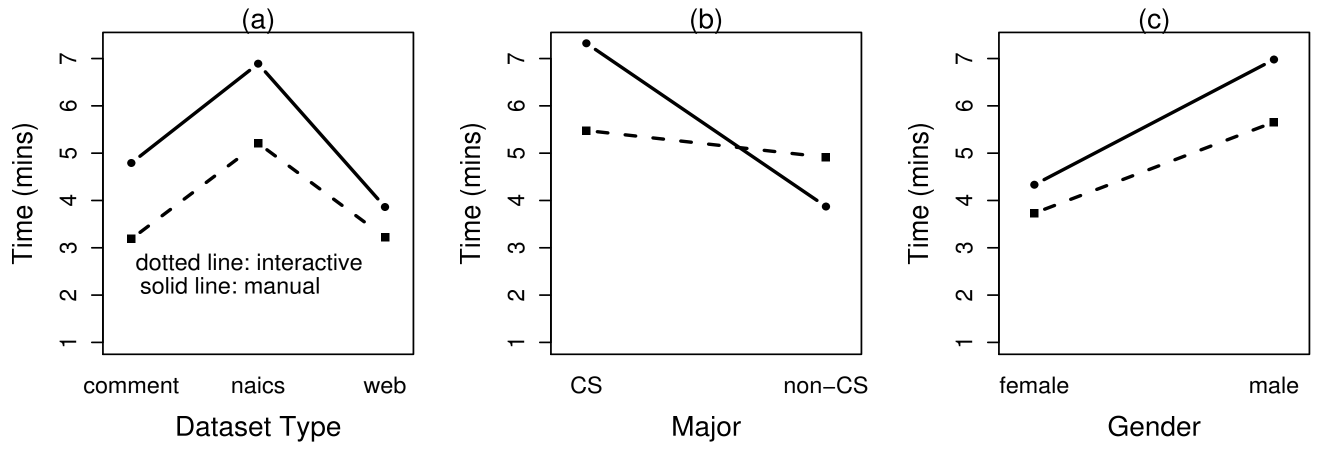


Figure 7.4: Interaction plot for mean construction time (in minutes).

7.3.3 Construction Time

Construction time is a good indicator for measuring the behavior of the participants and the characteristics of the datasets. The twenty-minute time limit we set for each concept hierarchy is a very generous upper limit so that the participants can adequately show their preferences in constructing the concept hierarchies.

We perform three two-way ANOVA tests - "construction method vs. dataset type", "construction method vs. major", and "construction method vs. gender" - to analyze the differences in construction time. Figure 7.4 (a), (b), and (c) show the interaction plots for the ANOVA tests.

Figure 7.4 (a) confirms the statement we made in Section 6.4.6 that the NAICS datasets need statistically significantly more time to construct concept hierarchies for them than the Web and the public comment datasets ($p < .001$). The participants also spent more time when they constructed the concept hierarchies manually than interactively ($p < .001$). No interaction effect is found between dataset type and construction method.

Figure 7.4 (b) shows that participants who major in Computer Science spent statistically significant more time in constructing a concept hierarchy than participants who major in other subjects spent. Moreover, the crossing of the lines indicates that there is also a statistically significant interaction between majors and construction methods on construction

Table 7.3: Statistics for participants' editing activity.

	Add	Delete	Drag&Drop	Rename	Promote	Undo	*Total*
Total	132.0	44.0	11,756.0	133.0	167.0	40.0	*12272.0*
Per user average	5.5	1.8	489.8	5.5	7.0	1.6	*511.3*
Per user per dataset average	0.4	0.1	35.3	0.4	0.5	0.1	*37.0*
Manual run average	0.6	0.2	40.5	0.6	0.5	0.1	*42.5*
Interactive run average	0.2	0.1	31.1	0.2	0.5	0.1	*32.2*

time ($p < .05$). Particularly, when using the interactive method, participants from different majors did not show statistical significant difference in time; while using the manual method, participants majoring in Computer Science spent statistically significantly more time than participants from other majors. If they use the manual method, the difference in time between Computer Science students and non-CS students is as large as 3.5 minutes, or greater than 30%. For the interactive method, the difference can nearly be detected. We need a larger scale user study to confirm the results. Although the result is statistically significant, it is tested on just 24 participants. We think it is necessary to use a larger scale user study and more strictly controlled study to draw a conclusion about major difference.

Figure 7.4 (c) shows that the male participants spent statistically significant more time on a concept hierarchy than the female participants ($p < .001$). Similar to major difference, we think it is necessary to use a larger scale user study to draw a conclusion about gender difference.

7.3.4 Number of Edits

The participants' editing activity is an interesting aspect to study. The types of editing that a participant can perform when constructing concept hierarchies include "adding a node", "deleting a node", "dragging and dropping a node", "renaming a node", "promoting a node", or "undoing" the previous actions.

Table 7.3 shows the total number of edits for each editing type, the average number of edits of each editing type per participant, the average number of edits of each editing type per participant per dataset, the average number of edits of each editing type per participant per dataset for the manual runs, and the average number of edits of each editing type per participant per dataset for the interactive runs.

The table shows that "drag & drop" is the most popular editing function. More than

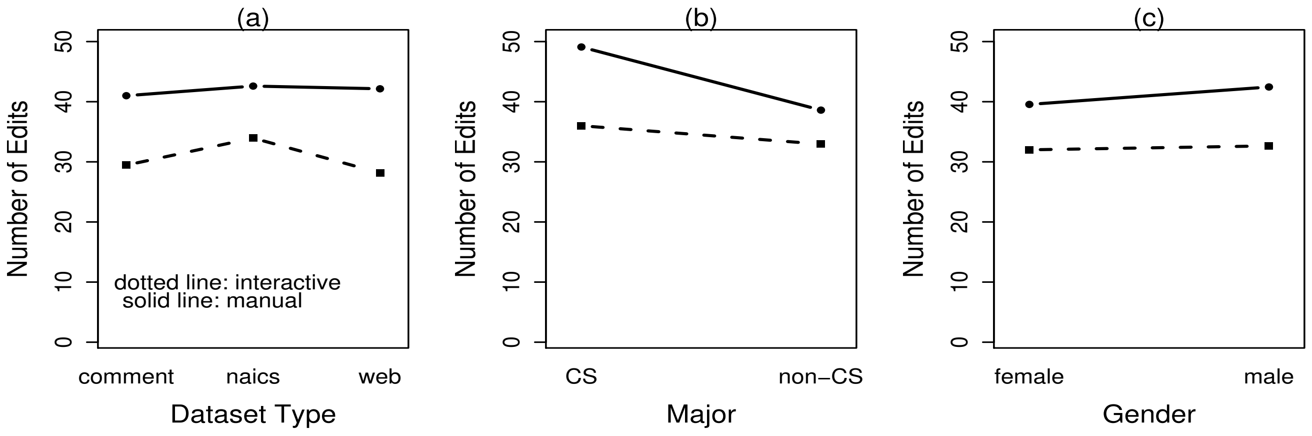


Figure 7.5: Interaction plot for mean number of edits.

90% of time, the participants used the "drag & drop" function to move the concepts around to construct the concept hierarchies. We believe that it is due to the easiness of using this function and the familiarity to this function through their past computer use experience.

Although all participants used the "drag & drop" function most frequently, they still show different editing habits. When we study the number of edits each participant made to build a concept hierarchy, we find statistical significant differences between the participants and between the datasets. We perform three two-way ANOVA tests as what we did for other dependant variables. Figure 7.5 (a), (b), and (c) show the ANOVA interaction plots for the tests of "construction method vs. dataset type", "construction method vs. major", and "construction method vs. gender" accordingly.

Figure 7.5 (a) confirms our findings in Section 6.4.6. It indicates that there is a statistically significant correlation between the number of edits and construction method ($p < .001$). Interactive runs use statistically significantly less number of edits than manual runs. No statistically significant correlation is found between number of edits and dataset type. There is a slight interaction effect between construction method and dataset type ($p < .01$). Specifically, when using the interactive method, the participants spent more edits on the NAICS datasets than the other two types of datasets. This trend did not repeat when they used the manual method.

Figure 7.5 (b) shows that there is no statistically significant correlation between the number of edits and major.

Figure 7.5 (c) shows that there is no statistically significant correlation between the number of edits and gender.

Considering the study reported in Section 7.3.3, when a dataset is less familiar to the participants, they spent statistically significant more time working on it, however, they did not generate statistically significantly more edits for it. If they did not spend the time on editing, they must have spent the time on thinking where to put a concept or how to organize the concepts. This is an interesting finding. Section 7.4 gives an explanation for this phenomenon.

7.3.5 Perceived Quality of Edits

The participants interacted with OntoCop to construct concept hierarchies. Both the human and the system made edits to the concept hierarchies. As described in Section 6.4.5, we asked the participants to evaluate the system's learning ability by rating the edits suggested by the system. We find that the ratings of system-generated edits are correlated with dataset types. We also asked the participants to evaluate their own edits by providing a 5-point scale rating: "very good"(5), "good"(4), "fair"(3), "bad"(2), and "trash"(1). This section studies perceived quality of the participants' own edits. We perform three two-way ANOVA tests of "construction method vs. dataset type", "construction method vs. major", and "construction method vs. gender" for perceived edit quality. Figure 7.6 (a), (b), and (c) illustrate the interaction plots for the tests.

From these tests, we only find a statistical significant correlation between perceived quality of participants' edits and dataset type ($p < .001$). Figure 7.6 (a) shows the interaction plot for the ANOVA test for "construction method vs. dataset type". From this figure we find that when dataset type is NAICS, the perceived quality of edits is statistically significant lower than when dataset type is public comments or the Web. No other effects are found for construction method, major, or gender.

We note that all perceived variables, including system learning ability (Section 6.4.5), quality of human edits (this section), and dataset difficulty (Section 7.3.2), depend and only depend on dataset type. The participants' demographics, and even the construction methods do not impact on these perceived dependent variables. We hypothesize that this may relate

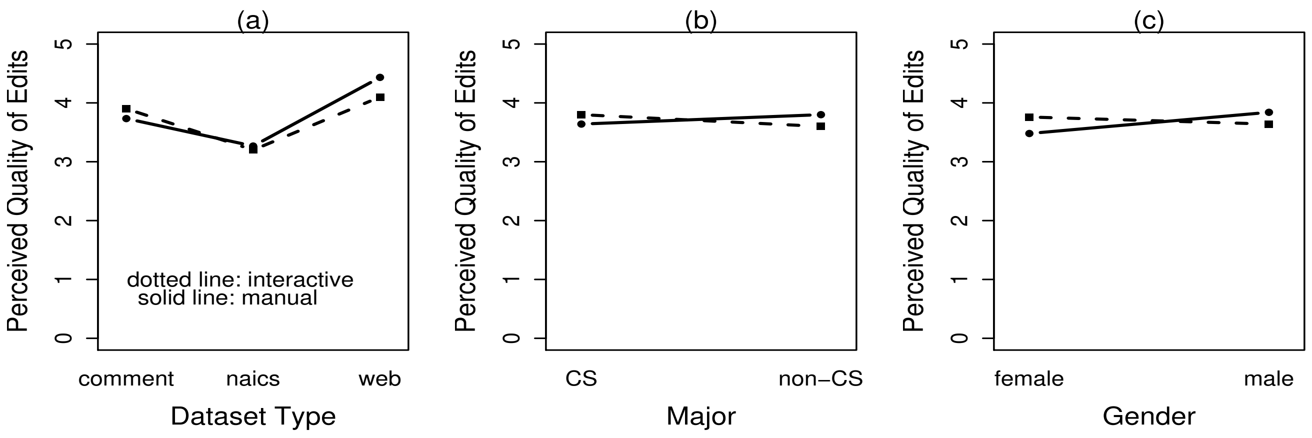


Figure 7.6: Interaction plot for mean perceived quality of edits.

to the fact that dataset type decides dataset difficulty; and dataset difficulty decides the other perceived variables.

7.4 Feature Use

We use heterogenous semantic features in our system. As described in Section 5.4, the semantic features include lexico-syntactic patterns, co-occurrence, contextual features, syntactic dependency features, definition and word length. In this section, we examine the top weighted features when each participant used OntoCop to construct concept hierarchies.

We obtain the top weighted features in the following steps:

1. *Obtain weights for features.*

   For each dataset D_i, participant j interacted with OntoCop and provided manual guidance $M_{ij}^{(k)}$ at the k^{th} iteration. The system learned a weight matrix $W_{ij}^{(k)}$ from $M_{ij}^{(k)}$ as described in Chapter 6. We obtain the diagonal vector $w_{ij}^{(k)}$ from $W_{ij}^{(k)}$. $w_{ij}^{(k)}$ contains the weights for each feature function for the k^{th} iteration when participant j constructed dataset D_i, where $i = 1, ..., 20$ and $j = 1, ..., 24$.

2. *Average the weights.*

   With the weight vectors, we can obtain an averaged weight vector for each participant.

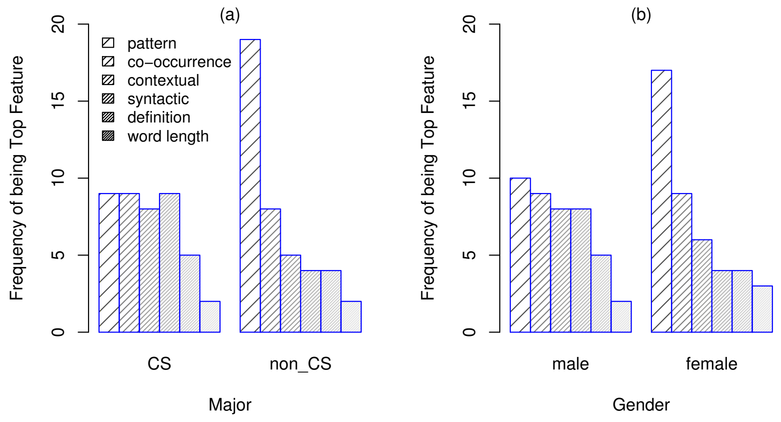


Figure 7.7: Use of different features.

Basically, we average all the weight vectors for a participant over all the datasets and all the iterations she interacted with OntoCop. The averaged weight vector $\bar{w_j}$ for participant j is:

$$\bar{w_j} = \frac{1}{20}\frac{1}{r_{ij}}\sum_{i=1}^{20}\sum_{k=1}^{r_{ij}} w_{ij}^{(k)}$$

where r_{ij} is the max number of iterations when participant j constructed concept hierarchy for D_i.

3. *Pick the top weighted features for each person.*

   For each participant j, we pick the two top weighted features f_{j1} and f_{j2} with the highest values in $\bar{w_j}$.

4. *Count votes for top weighted features.*

   For each feature f_t, we count how many participants used it as a top feature. Moreover, we can count how many participants in a group used it as a top feature. $t =$ (”pattern”, ”co-occurrence”, ”contextual”, ”syntactic”, ”definition”, ”word length”).

$$\forall f_t, \forall j \in \text{a group},$$

$$f_t\text{'s frequency of being a top feature} = Count(f_t == f_{j1}, \text{ or } f_t == f_{j2}).$$

Figure 7.7 (a) plots the top weighted features for CS-majors and non-CS majors. The plot shows that CS-majors used a variety of features equally likely, while non-CS majors used lexico-syntactic patterns much more frequently than other types of features.

Figure 7.7 (b) plots the top weighted features for males and females. It shows that males used different types of features equally likely, while females used patterns much more frequently than other types of features.

This is an interesting finding. The patterns are easy to use to quickly identify relations; while other types of features are more complex to be evaluated by a human. We do not claim that users make judgements based on the list of features provided in our system. However, the finding suggests that non-CS majors and females may correlate with a simpler, pattern-based strategy, whereas CS-majors and males correlate with a more complex strategy to organize information.

The different use pattern of underlying feature functions, might be the reason why people construct different concept hierarchies for the same dataset and why people show different user behaviors such as construction time and number of edits. In Section 7.3.3 and Section 7.3.4, we find that although CS-majors and males spent more time, they did not make more edits. Considering they may use a more complex strategy, this becomes understandable because they might think more and hence spent longer time between each edit. Nonetheless, we believe it needs more extensive user study to confirm this interesting hypothesis.

7.5 Self-agreement

The user study consisted of two phases, an initial phase and an repeat phase. At the end of the initial phrase, one of the questions in the after-study questionnaire is "*Would you like to be invited to repeat the study a few weeks later?*". Among those participants agreed to come back, we randomly invited 12 of them to join the repeat phase three weeks after the initial phase. The three week period ensured that the participants only have limited memory of the details about the tasks performed in the initial phase. The details of the participants in the repeat phase is shown in Table 7.4.

The participants in the repeat phase were asked to organize concept hierarchies for the same set of datasets and using the same order of construction methods as in the initial phase.

Institutes	
Carnegie Mellon University	7
University of Pittsburgh	5
Gender	
Male	6
Female	6
Majors	
Arts	1
Chemical Engineering	1
Cognitive Science	2
Computer Science	4
Public Policy	2
Psychology	2
English Proficiency	
Native speaker	8
Non-native speaker (with high proficiency)	4

Table 7.4: Statistics of participants in the repeat phase.

we study self-agreement to find out if the variation among users is really due to personal preferences instead of random variations. Particularly, we use Fragment-Based Similarity (FBS) (Section 3.5) to calculate the similarity between the concept hierarchies constructed in the initial phase and the corresponding concept hierarchies constructed in the repeat phase by the same participant.

Table 7.5 shows the self-agreement measured in FBS. The first row indicates the maximum, minimum, and average self-agreement between concept hierarchies for any participant and any dataset. The max self-agreement is as high as 1, which was a participant who interactively constructed a concept hierarchy for the "wolf" dataset. The minimum self-agreement is 0.37; the average self-agreement is 0.74, which is high.

The second row shows that the participant who was most self-agreeable could produce a self-agreement of 0.81 on average across different datasets; the participant who was least self-agreeable could produce a self-agreement of 0.63. The average is still 0.74.

The third row shows that for a dataset that produces the highest self-agreement among all the participants, the self-agreement value is 0.95. For a dataset that produces the least

Table 7.5: The maximum, minimum, and average Self-agreement values; measured in FBS.

Self agreement (in FBS)	Max	Min	Average
per participant per dataset	1	0.37	0.74
per participant	0.81	0.63	0.74
per dataset	0.95	0.62	0.74
manual runs	0.98	0.37	0.73
interactive runs	1	0.45	0.76

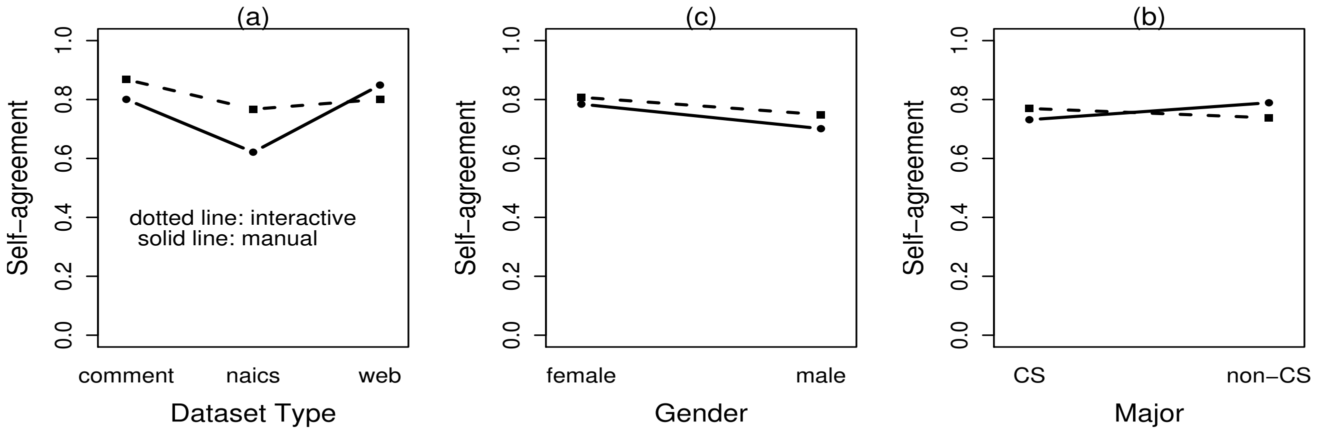


Figure 7.8: Interaction plot for self-agreement; measured in FBS.

self-agreement among all the participants, the self-agreement value is 0.62. The average is still 0.74 for all datasets.

The fourth and the fifth rows illustrate the max, min, and average self-agreement for the manual and the interactive runs, respectively. The max, min, and average self-agreements of the interactive runs are a bit higher than the manual runs.

These self-agreement values are all at the high end of FBS, which shows that the participants are quite self-consistent when constructing concept hierarchies at different times.

In order to understand what the influencing factors are for self-agreement, we perform the three two-way ANOVA tests for self-agreement. Figure 7.8 (a) show that there is a statistically significant correlation between self-agreement and dataset type ($p < .001$). More difficult datasets, i.e. the NAICS datasets, yield lower self-agreement for a participant and hence less consistent concept hierarchies, but it is still very good.

Figures 7.8 (a), (b), and (c) show that there is a slight correlation between self-agreement and construction method ($p < .05$). Using the interactive method yields higher self-agreement as compared to using the manual method. There are no other effects caused by major or gender.

Note that there were only 12 participants in the repeat phase. Although some results about self-agreement are statistically significant, they are based on a small and somewhat uniform user population (college students). We plan to extend the user study to a larger scale evaluation.

7.6 Summary

This chapter reports the experiments and analysis for a user study in concept hierarchy construction. We explore the commonality and differences between the concept hierarchies that constructed by different people. We find that every dataset has a part which most participants agree on; every dataset also has a part which no one agrees on with each other. In this user study, we emphasis on understanding what the differences are and why they are different. We find several possible influencing factors, including dataset type, construction method, major, and gender of the participants. We then evaluate these factors on various aspects of concept hierarchy construction through statistical significance tests.

Through the user study, we find that people are quite self-agreeable to themselves when constructing concept hierarchies (Section 7.5). This novel finding provides a foundation to study the personal preferences among people.

Moreover, we find that construction method is an important factor for concept hierarchy construction. OntoCop's interactive function helps concept hierarchy construction in several aspects. Participants used much less time (Section 7.3.3) and much less edits (Section 7.3.4) to construct concept hierarchies when using the interactive method as compared to using the manual method. Using OntoCop's interactive functions, the dataset also appear less difficult to the participants (Section 7.3.2). Moreover, interactive runs help the participants to be more self-consistent (Section 7.5).

We also find that dataset difficulty, implied by dataset type, is another important indicator for both the system's and the user's performance. When a dataset is more difficult, both the user and the system may do poorly in various aspects of concept hierarchy construction. The reason is mainly due to the user's lack of prior knowledge or unfamiliarity with the concepts and the relations in the domain. However, it maybe because the participants in this user study were college undergraduates or graduates, they share similar vocabulary and are familiar with the Web and emails, but not familiar with the industry standards (NAICS).

In addition, we find that people with different demographics show different feature use patterns (Section 7.4). Feature use patterns could be a reason why people use different amount of time and make different amount of edits to a concept hierarchy. However, further investigation needs to be done to draw conclusions about gender difference and major difference in feature use.

Chapter 8

Conclusion

This chapter concludes the dissertation by summarizing the research in Section 8.1 and highlighting the contribution in Section 8.2, followed by a discussion of a few future directions in Section 8.3.

8.1 Research Summary

This dissertation studies how to effectively organize information and how to encode user-specific or task-specific preferences in the organizing process. To "organize information", we present concept extraction (Chapter 4) and metric-based concept hierarchy construction (Chapter 5). To "encode preferences", we present human-guided concept hierarchy construction (Chapter 6). We also study how to evaluate user behaviors during concept hierarchy construction (Chapter 7) and how to compare concept hierarchy similarity (Section 3.5). In this section, we summarize the dissertation research as follows.

Metric-based concept hierarchy construction is a novel automatic concept hierarchy construction framework. It constructs an initial concept hierarchy from data and presents it to the user. Through an analysis of how people build a concept hierarchy step by step, the framework mimics the steps and turns concept hierarchy construction into an incremental clustering process. In this process, every step of adding a new concept into the concept hierarchy is transformed into an optimization problem. The optimization is based on minimum evolution of concept hierarchy structure and semantic distances, modelling of concept abstractness, and modelling of concept coherence. For each pair of concepts that have an immediate relation, their semantic distance is modeled as an integration of many semantic feature functions. Each feature is carefully chosen and corresponds to a state-of-the-art technique. Therefore, this framework provides a general platform to include multiple state-of-the-art techniques, and find the best weights for each technique through the optimization. As a result, metric-based concept hierarchy construction generates initial concept hierarchies with good quality.

Incorporating personal preferences in concept hierarchy construction is challenging. The human-guided concept hierarchy construction framework allows a user to provide periodic manual guidance and interacts with a learning algorithm to produce a concept hierarchy. Through human-computer interaction, the human and the machine work together to organize concepts into concept hierarchies. The user interfaces of such systems are required to be user-friendly. OntoCop, our interactive concept hierarchy construction tool, satisfies such requirements. Its interface captures how a user organizes the concept hierarchy and the learning algorithm translates this information into matrices that can be easily adopted. The algorithm uses the manual guidance represented by these matrices to train new concept hierarchy construction models which adapt to the user's preferences of how to organize the concept hierarchy. The model is then used to make predictions of how to further construct the concept hierarchy according to this particular user's preferences. In this way, the user is successfully put-into-the-loop and able to build *personal* concept hierarchies.

To evaluate the system effectiveness and to study how to evaluate hierarchy similarity, we propose a novel metric - Fragment-Based Similarity (FBS) - which employs a unique bag-of-word representation for concept hierarchies and evaluates the similarity between concept hierarchies fragment by fragment. The dissertation empirically evaluates various design decisions that lead to this new metric. FBS can be an very efficient similarity measure for hierarchies in general. It well approximates Tree Edit Distance, however greatly improves Tree Edit Distance's efficiency from NP-hard to a time complexity of only $O(n^3)$ ($O(n)$ if pairwise node similarities are pre-calculated).

We conduct a user study involving 24 participants to evaluate whether human-guided concept hierarchy construction can successfully assist people to construct concept hierarchies that reflect their personal preferences, and whether the interactive system is able to accelerate the process as compared to constructing the concept hierarchies manually. The users are asked to evaluate the system's predictions on-the-fly during each human-computer interaction. The results show that our system achieves a high prediction accuracy (above 92%). The time and edits used to construct a concept hierarchy are also greatly reduced by 20% to 30%. The user study demonstrates that human-guided concept hierarchy construction is able to generate concept hierarchies with manually-built quality and with much more efficiency.

Besides system effectiveness and efficiency, we also analyze user behaviors during concept hierarchy construction in the user study. In particular, we explore what are the dataset-specific or user-specific differences in the concept hierarchies that people construct, whether people are self-consistent, what are the influencing factors for producing different concept hierarchies, and how these factors interact with different construction methods. We take an exploratory approach to study the collected data, including user demographics, concept hierarchies being constructed, editing logs, and answers to the questionnaires. Through a clustering of users and concept hierarchies, we discover several possible influencing factors. Based on them, we conduct statistical significance tests to identify the real influencing factors that affect people on constructing different concept hierarchies. We find that dataset difficulty is a major factor affecting how people organize information into concept hierarchies. Other factors, such as major, gender, and feature use patterns, need further experiments to draw firm conclusions. We also find out that people are quite self-consistent in building concept hierarchies when they are asked to construct concept hierarchies for the same topic at different times. This novel finding provides foundations to study difference in concept hierarchy construction behaviors between different individuals.

8.2 Significance of the Dissertation

This dissertation addresses how to construct concept hierarchies from text collections both automatically and with a-human-in-the-loop. It integrates techniques in machine learning, natural language processing, and information retrieval into one framework to advance the research of concept hierarchy construction. Moreover, this work not only just integrates techniques from various research fields, but also contributes to the development of those fields. Particularly, it has a significant impact on research areas including *concept hierarchy construction*, *human-computer interaction*, and *hierarchy similarity measurement*.

One of the major contributions of this dissertation research is the *use of heterogenous semantic features*. Traditionally, researchers in knowledge acquisition or concept hierarchy construction only use a single type of techniques in their work. Examples include lexico-syntactic patterns, word co-occurrences, or syntactic dependency features. Each technique usually requires a unique way to be applied on data. For example, patterns need to be matched with instances in text and usually are used together with the bootstrapping technique; syntactic dependency features require splitting documents into sentences and parsing the sentences before generating the features. Therefore, by default, different features do not appear in a uniform format. Even if we can represent them all in numeric numbers, how to incorporate multiple features to decide a concept hierarchy's structure is still a problem. Researcher has managed to apply patterns first then word co-occurrence statistics (e.g., PMI) to control quality of patterns and instances. However, such an add-modules-one-by-one model cannot be continued once the number of features goes up. This is one of the main reasons why researchers usually just use one technique for concept hierarchy construction instead of taking multiple perspectives. Instead of combining techniques using ad-hoc methods, in this dissertation research we design a general framework to support multiple techniques. Particularly, we combine all techniques into a semantic feature vector and use the vector to calculate a semantic distance between concepts. Through optimizing the overall semantic distance among concepts, we optimize the concept hierarchy structure to ensure that a concept hierarchy is organized based on a sensible guideline. Through this process, each feature's weight is optimized. Our research moves beyond the limitation of traditional use of features and incorporates heterogeneous semantic evidence into the learning process. This is a significant contribution to concept hierarchy construction. Moreover, we can flexibly add or reduce features, and study how each feature contributes to concept hierarchy construction under various situations.

Besides optimization of concept hierarchy structure, we employ two more optimization strategies for concept hierarchy construction. Both strategies are inspired by our observation of concept hierarchies' characteristics. The first is *modeling of concept abstractness*. Specifically, we perform different modeling for concepts at different horizontal levels of a concept hierarchy to ensure that abstract concepts and concrete concepts are handled differently. The second is *modeling of concept coherence*. It aims to solve the inconsistency issue caused by concept insertions which are only based on immediate/local pair-wise relations. If we don't deal with concept coherence, oftentimes an concept hierarchy construction algorithm produces inconsistent vertical chain of concepts in a concept hierarchy. The problems addressed by these two strategies have actually been noticed by researchers in Computational Linguistics and Information Retrieval [SC99]. However, researchers have not yet designed new algorithms to handle them. We are fortunate enough to be the first to use statistical machine learning and optimization techniques to approach these problems. We expect that more followup research will emerge and a variety of approaches will be proposed to explore these issues.

A very important factor in concept hierarchy construction is the human. In this dissertation research, we put much effort in studying how to incorporate the human seamlessly in the process of concept hierarchy construction to bring personality to the static, machine-generated concept hierarchies. From another perspective, the machine must seamlessly help the human organize information into concept hierarchies with no or little interruption to user experience. OntoCop is such an easy-to-use software tool, in which people can easily drag and drop concepts around to organize information. It captures human actions and uses them as guidance to train statistical learning models and predict organizations of other concepts. In this way, the system greatly reduces manual efforts for concept hierarchy construction and realizes real-time interactive concept hierarchy construction.

For any empirical and experimental research, evaluation is always very important. Concept hierarchy is not an exception. However, because it is a personalized task, it seems that only the person who creates/uses the concept hierarchy has the authority to judge whether a concept hierarchy is good or bad. This constraint greatly increases the difficulty to evaluate the effectiveness of concept hierarchy construction. In this dissertation research, we adopt several methods to evaluate concept hierarchies. One method is to let the user to directly evaluate the system. This includes a subjective evaluation by questionnaires and an on-the-fly evaluation on machine-generated concept pairs during each human-computer interaction. Another method is to compare with existing concept hierarchies when we use the proposed approach to re-construct them. This kind of comparison to community opinions should not be used as the final judge for a concept hierarchy, however, it provides good references. Comparing with existing concept hierarchies is a problem of comparing hierarchy similarity. So far, this problem has no sufficiently efficient solution. Based on concept hierarchies' characteristics and our observation over how people compare hierarchies, we propose a novel hierarchy similarity measure, Fragment-Based Similarity (FBS). FBS well approximates Tree Edit Distance but greatly reduces its complexity from NP-hard to polynomial time. We expect FBS to be widely used in various applications where there exist needs to compare hierarchies. We also welcome researchers to propose other methods based on a fragment view of hierarchy structure.

In summary, the research in this dissertation is the first step of *concept hierarchy construction*, and an important step forward of *concept hierarchy construction*. This dissertation research addresses important problems of concept hierarchy construction, especially considering how to better model the problems with good theoretical foundations, to study the problems via extensive empirical experiments and user studies, and to solve the problems by developing practical applications for constructing concept hierarchies. It develops both automated and interactive methods that assist information seeking, organization, and management activities. Methods proposed in the research will not only lead to practical systems of immediate benefit for users, but also enhance our ability to reason about the sophisticated information systems of the future. The better theoretical foundation, the more extensive empirical experiments and user studies, and the better modeling of various aspects of concept hierarchy construction in this new research show a bright future of *concept hierarchy construction*.

8.3 Future Directions

Research on *concept hierarchy construction* is still in its infancy. The theoretical and conceptual challenges are deep and exciting. The following are descriptions of some future directions of this new research.

8.3.1 Interactive Concept Suggestion

In human-guided concept hierarchy construction, the system and the user work together to create a concept hierarchy through interactions. The system models manual guidance provided by the user and learns the user preferences from it. The system then organizes the unmodified concepts and updates the concept hierarchy based on the manual guidance. Thus the organization of the concepts are updated in real-time from iteration to iteration according to the user preferences. In this process, we assume that the concepts are fixed and have been acquired by concept extraction. This separation of concept acquisition and relation acquisition simplifies the task. However, it is not the most desirable method because not only the organization needs to be customized, but also the concepts in a concept hierarchy need to be customized based on the user's guidance which arrives in real-time during the human-computer interactions.

Although in the first step of concept extraction (Chapter 4), we have extracted almost all possible concept candidates, only a few dozens to a few hundreds are kept in the concept set and presented to the user. Some useful concepts might be thrown away during concept filtering and concept unification, therefore at the end of concept extraction, we only reach a recall around 60%. Such a recall value is not bad at all for a task that cares more about precision (e.g., Web search), however it might be a little bit low for an information organization task which cares about both precision and recall. Another reason for this low recall is that users create many self-defined concepts in the interactive process. These self-defined concepts are about the domain, however do not explicitly appear in the document collection. Therefore it is difficult to discover them from the collection itself.

A new mechanism of interactive concept suggestion is an interesting extension to this dissertation research. During human-computer interactions, both the concept sets and the organization of these concepts should be updated by both the user and the system. When a user adds a concept or deletes a concept in the concept hierarchy, she actually indicates her interest of topics for the domain and for the task. Based on the concepts being added or deleted, we should be able to predict and introduce new concepts that the user is interested from the document collection and external resources such as the Web. Techniques such as query suggestion and natural language generation should be reviewed and new solutions should be explored for the task of interactive concept suggestion in concept hierarchy construction.

8.3.2 Multiple Inheritance

In Chapter 1, we argue that the best form of organizing information is multiple tree representations with additional links between nodes in a tree. In this dissertation research, we address this issue by having the users to participate in an interactive process and taking into account their personal preferences in concept hierarchy construction so that each concept hierarchy is one tree with one specific view to the data. However, we did not study how to allow additional links in a concept hierarchy.

In order to introduce additional links into a tree, we must support multiple inheritance. Some concepts could have multiple parents, such as "bank" is a "financial institute" and it is also a part of a "river". The current framework only chooses a single "best" position for each concept. In the future work, we will allow some concepts to be positioned in multiple positions in the concept hierarchy. In theory, this can be done within the framework by relaxing the constraints when assigning a single position for a concept. This relaxation will probably incur more computational cost. We hence must make careful decisions on two issues: which concepts should be positioned in more than one locations and where are the best locations. These decisions can be made in different ways.

We could first adopt a heuristic approach by setting a threshold on the changes that a concept brings into a concept hierarchy. Recall that we adopt a minimum evolution assumption that a concept hierarchy grows in a way that minimum changes to its structure and minimum increases to its overall distances should be preserved. If the change of the overall semantic distances caused by an insertion is less than a pre-determined threshold, the position where a concept is inserted into will be considered as valid for this concept. This method is simple and relatively efficient. However, some concepts may introduce less changes into the concept hierarchy's overall structure, others may introduces more changes to it. Hence it might not be trivial to set a good and general threshold.

Alternatively, we can train classification models to predict if a concept should be positioned into multiple locations in the concept hierarchy or not. Given a gold standard concept hierarchy, we can extract features to determine if a concept should have multiple parents or multiple children or neither. In general, this method could be more robust than the above heuristic method, but it requires more computational cost and training data to achieve reasonable performance.

8.3.3 Study of User Behaviors

The user study presented in Chapter 7 provides a preliminary study of the impact of participants' demographic differences and feature use differences in constructing a concept hierarchy. Among them, we are especially excited to explore whether and how different feature use patterns might be the reason why people construct concept hierarchies differently.

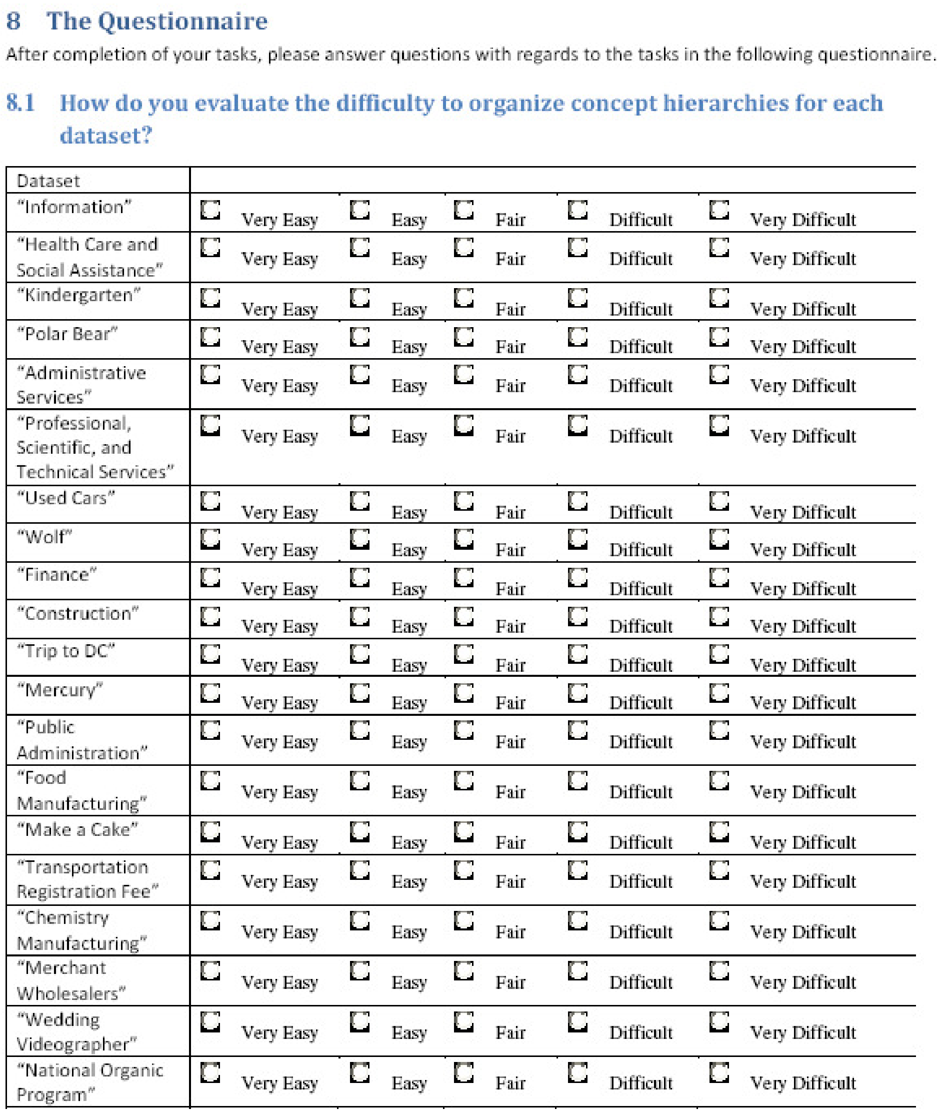
In the preliminary user study, we discover that some users such as females and non-CS majors tend to often use lexico-syntactic patterns in their decision-making process to quickly infer relations between concepts, and other users such as males and CS majors tend to often take a more complex strategy to use many features together to identify relations. This makes us think that it might not be the gender difference nor the major difference that yield the differences in concept hierarchies that people create. It might be the feature use patterns. In another word, we suspect that people who mainly use patterns probably produce different concept hierarchies than people who use a combination of many features. If we can prove the hypothesis that different user groups tend to use one or more specific features to organize information, that will be a very interesting and important finding. The results may have many implications and we can take advantage of this knowledge to better serve a user.

For instance, in the concept hierarchy construction process, for a user who tends to use patterns only, we can increase the weights for her pattern-based features; for a user who tends to use a variety of features, we can constrain the variance of the weights for all features to remain within a limited range. Using this more targeted strategy based on user groups, the system could achieve the concept hierarchy in a user's mind faster and better.

This may also be helpful for people who routinely need to organize information, e.g., government analysts and lawyers. Learning their general working preferences may enable the system to begin a new dataset with features that are tuned for an individual, rather than waiting for the individual to do some manual training.

Appendix A

Questionnaire

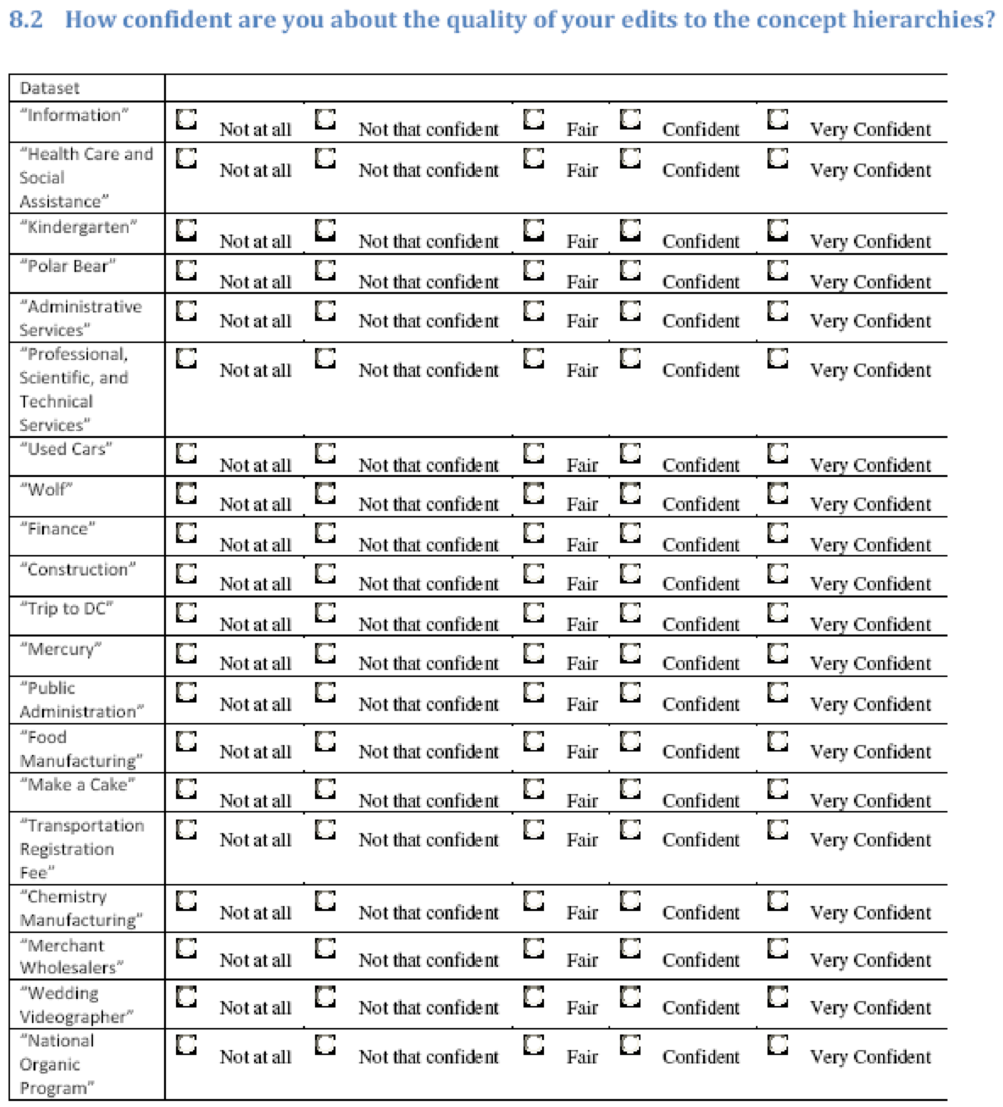


8 The Questionnaire

After completion of your tasks, please answer questions with regards to the tasks in the following questionnaire.

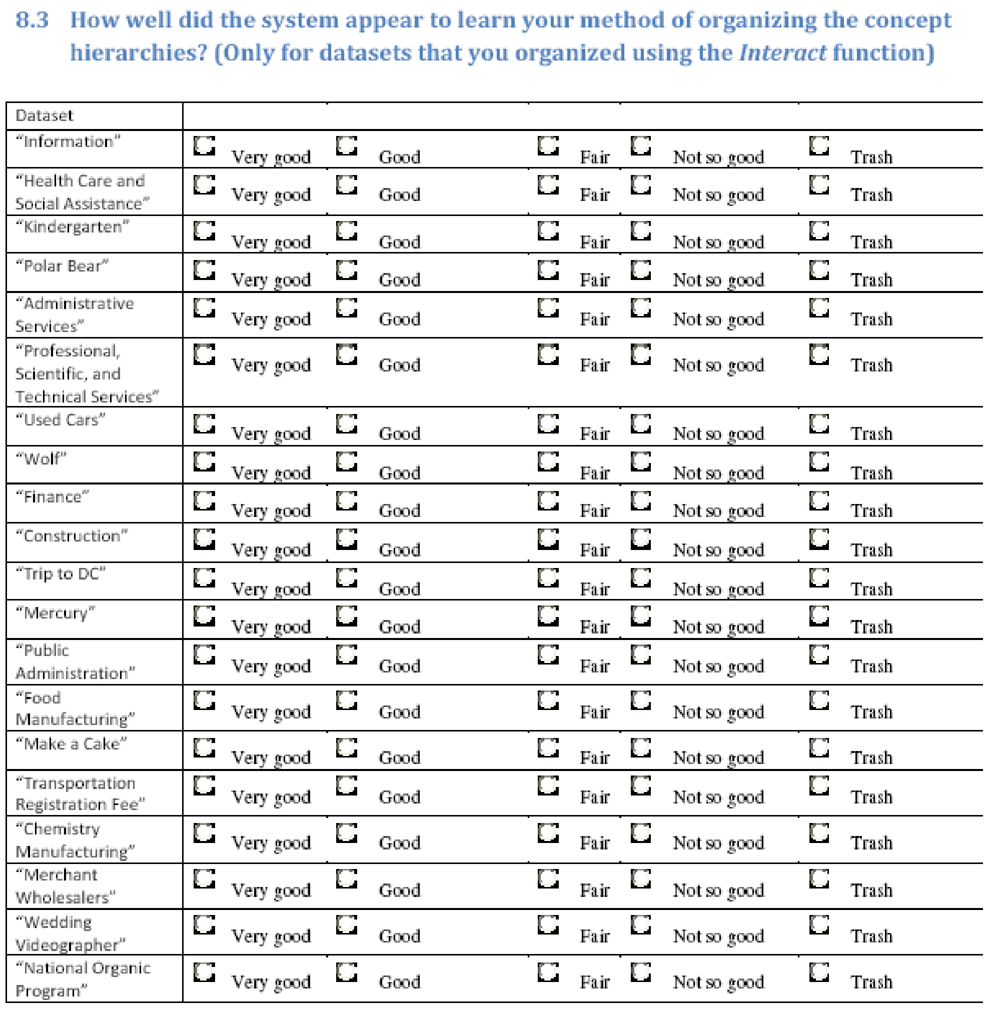
8.1 How do you evaluate the difficulty to organize concept hierarchies for each dataset?

Dataset					
"Information"	☐ Very Easy	☐ Easy	☐ Fair	☐ Difficult	☐ Very Difficult
"Health Care and Social Assistance"	☐ Very Easy	☐ Easy	☐ Fair	☐ Difficult	☐ Very Difficult
"Kindergarten"	☐ Very Easy	☐ Easy	☐ Fair	☐ Difficult	☐ Very Difficult
"Polar Bear"	☐ Very Easy	☐ Easy	☐ Fair	☐ Difficult	☐ Very Difficult
"Administrative Services"	☐ Very Easy	☐ Easy	☐ Fair	☐ Difficult	☐ Very Difficult
"Professional, Scientific, and Technical Services"	☐ Very Easy	☐ Easy	☐ Fair	☐ Difficult	☐ Very Difficult
"Used Cars"	☐ Very Easy	☐ Easy	☐ Fair	☐ Difficult	☐ Very Difficult
"Wolf"	☐ Very Easy	☐ Easy	☐ Fair	☐ Difficult	☐ Very Difficult
"Finance"	☐ Very Easy	☐ Easy	☐ Fair	☐ Difficult	☐ Very Difficult
"Construction"	☐ Very Easy	☐ Easy	☐ Fair	☐ Difficult	☐ Very Difficult
"Trip to DC"	☐ Very Easy	☐ Easy	☐ Fair	☐ Difficult	☐ Very Difficult
"Mercury"	☐ Very Easy	☐ Easy	☐ Fair	☐ Difficult	☐ Very Difficult
"Public Administration"	☐ Very Easy	☐ Easy	☐ Fair	☐ Difficult	☐ Very Difficult
"Food Manufacturing"	☐ Very Easy	☐ Easy	☐ Fair	☐ Difficult	☐ Very Difficult
"Make a Cake"	☐ Very Easy	☐ Easy	☐ Fair	☐ Difficult	☐ Very Difficult
"Transportation Registration Fee"	☐ Very Easy	☐ Easy	☐ Fair	☐ Difficult	☐ Very Difficult
"Chemistry Manufacturing"	☐ Very Easy	☐ Easy	☐ Fair	☐ Difficult	☐ Very Difficult
"Merchant Wholesalers"	☐ Very Easy	☐ Easy	☐ Fair	☐ Difficult	☐ Very Difficult
"Wedding Videographer"	☐ Very Easy	☐ Easy	☐ Fair	☐ Difficult	☐ Very Difficult
"National Organic Program"	☐ Very Easy	☐ Easy	☐ Fair	☐ Difficult	☐ Very Difficult



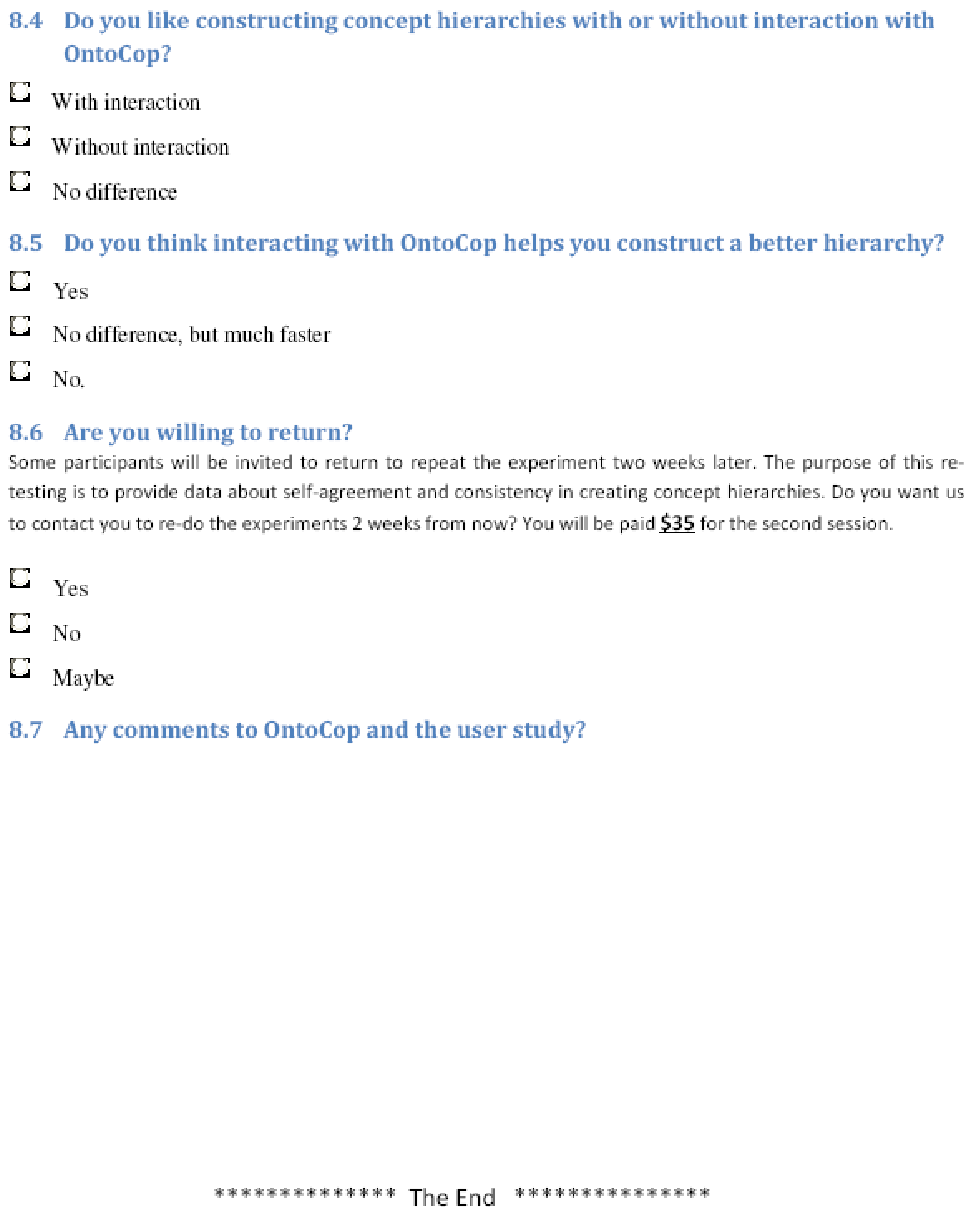
8.2 How confident are you about the quality of your edits to the concept hierarchies?

Dataset					
"Information"	☐ Not at all	☐ Not that confident	☐ Fair	☐ Confident	☐ Very Confident
"Health Care and Social Assistance"	☐ Not at all	☐ Not that confident	☐ Fair	☐ Confident	☐ Very Confident
"Kindergarten"	☐ Not at all	☐ Not that confident	☐ Fair	☐ Confident	☐ Very Confident
"Polar Bear"	☐ Not at all	☐ Not that confident	☐ Fair	☐ Confident	☐ Very Confident
"Administrative Services"	☐ Not at all	☐ Not that confident	☐ Fair	☐ Confident	☐ Very Confident
"Professional, Scientific, and Technical Services"	☐ Not at all	☐ Not that confident	☐ Fair	☐ Confident	☐ Very Confident
"Used Cars"	☐ Not at all	☐ Not that confident	☐ Fair	☐ Confident	☐ Very Confident
"Wolf"	☐ Not at all	☐ Not that confident	☐ Fair	☐ Confident	☐ Very Confident
"Finance"	☐ Not at all	☐ Not that confident	☐ Fair	☐ Confident	☐ Very Confident
"Construction"	☐ Not at all	☐ Not that confident	☐ Fair	☐ Confident	☐ Very Confident
"Trip to DC"	☐ Not at all	☐ Not that confident	☐ Fair	☐ Confident	☐ Very Confident
"Mercury"	☐ Not at all	☐ Not that confident	☐ Fair	☐ Confident	☐ Very Confident
"Public Administration"	☐ Not at all	☐ Not that confident	☐ Fair	☐ Confident	☐ Very Confident
"Food Manufacturing"	☐ Not at all	☐ Not that confident	☐ Fair	☐ Confident	☐ Very Confident
"Make a Cake"	☐ Not at all	☐ Not that confident	☐ Fair	☐ Confident	☐ Very Confident
"Transportation Registration Fee"	☐ Not at all	☐ Not that confident	☐ Fair	☐ Confident	☐ Very Confident
"Chemistry Manufacturing"	☐ Not at all	☐ Not that confident	☐ Fair	☐ Confident	☐ Very Confident
"Merchant Wholesalers"	☐ Not at all	☐ Not that confident	☐ Fair	☐ Confident	☐ Very Confident
"Wedding Videographer"	☐ Not at all	☐ Not that confident	☐ Fair	☐ Confident	☐ Very Confident
"National Organic Program"	☐ Not at all	☐ Not that confident	☐ Fair	☐ Confident	☐ Very Confident



8.3 How well did the system appear to learn your method of organizing the concept hierarchies? (Only for datasets that you organized using the *Interact* function)

Dataset					
"Information"	☐ Very good	☐ Good	☐ Fair	☐ Not so good	☐ Trash
"Health Care and Social Assistance"	☐ Very good	☐ Good	☐ Fair	☐ Not so good	☐ Trash
"Kindergarten"	☐ Very good	☐ Good	☐ Fair	☐ Not so good	☐ Trash
"Polar Bear"	☐ Very good	☐ Good	☐ Fair	☐ Not so good	☐ Trash
"Administrative Services"	☐ Very good	☐ Good	☐ Fair	☐ Not so good	☐ Trash
"Professional, Scientific, and Technical Services"	☐ Very good	☐ Good	☐ Fair	☐ Not so good	☐ Trash
"Used Cars"	☐ Very good	☐ Good	☐ Fair	☐ Not so good	☐ Trash
"Wolf"	☐ Very good	☐ Good	☐ Fair	☐ Not so good	☐ Trash
"Finance"	☐ Very good	☐ Good	☐ Fair	☐ Not so good	☐ Trash
"Construction"	☐ Very good	☐ Good	☐ Fair	☐ Not so good	☐ Trash
"Trip to DC"	☐ Very good	☐ Good	☐ Fair	☐ Not so good	☐ Trash
"Mercury"	☐ Very good	☐ Good	☐ Fair	☐ Not so good	☐ Trash
"Public Administration"	☐ Very good	☐ Good	☐ Fair	☐ Not so good	☐ Trash
"Food Manufacturing"	☐ Very good	☐ Good	☐ Fair	☐ Not so good	☐ Trash
"Make a Cake"	☐ Very good	☐ Good	☐ Fair	☐ Not so good	☐ Trash
"Transportation Registration Fee"	☐ Very good	☐ Good	☐ Fair	☐ Not so good	☐ Trash
"Chemistry Manufacturing"	☐ Very good	☐ Good	☐ Fair	☐ Not so good	☐ Trash
"Merchant Wholesalers"	☐ Very good	☐ Good	☐ Fair	☐ Not so good	☐ Trash
"Wedding Videographer"	☐ Very good	☐ Good	☐ Fair	☐ Not so good	☐ Trash
"National Organic Program"	☐ Very good	☐ Good	☐ Fair	☐ Not so good	☐ Trash



8.4 Do you like constructing concept hierarchies with or without interaction with OntoCop?

- ☐ With interaction
- ☐ Without interaction
- ☐ No difference

8.5 Do you think interacting with OntoCop helps you construct a better hierarchy?

- ☐ Yes
- ☐ No difference, but much faster
- ☐ No.

8.6 Are you willing to return?

Some participants will be invited to return to repeat the experiment two weeks later. The purpose of this re-testing is to provide data about self-agreement and consistency in creating concept hierarchies. Do you want us to contact you to re-do the experiments 2 weeks from now? You will be paid **$35** for the second session.

- ☐ Yes
- ☐ No
- ☐ Maybe

8.7 Any comments to OntoCop and the user study?

************* The End ***************

Bibliography

[ADMR05] David Aumueller, Hong H. Do, Sabine Massmann, and Erhard Rahm. Schema and ontology matching with COMA++. In *Proceedings of the 24th ACM SIGMOD International Conference on Management of Data (SIGMOD 2005)*, pages 906–908, 2005.

[AFB03] James Allan, Ao Feng, and Alvaro Bolivar. Flexible intrinsic evaluation of hierarchical clustering for tdt. In *Proceedings of the 12th International Conference on Information and Knowledge Management (CIKM 2003)*, 2003.

[ATDE09] Eytan Adar, Jaime Teevan, Susan T. Dumais, and Jonathan L. Elsas. The web changes everything: understanding the dynamics of web content. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining (WSDM 2009)*, pages 282–291, 2009.

[BC99] Matthew Berland and Eugene Charniak. Finding parts in very large corpora. In *Proceedings of the 27th Annual Meeting for the Association for Computational Linguistics (ACL 1999)*, 1999.

[BDKW07] Horst Bunke, Peter J. Dickinson, Miro Kraetzl, and Walter D. Wallis. A graph-theoretic approach to enterprise network dynamics. *Information Retrieval, Khauser, Boston, MA*, 2007.

[BE08] Michele Banko and Oren Etzioni. The tradeoffs between open and traditional relation extraction. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics with the Human Language Technologies Conference (ACL/HLT 2008)*, 2008.

[Bha06] Rajendra Bhatia. *Positive definite matrices (princeton series in applied mathematics)*. Princeton University Press, December 2006.

[Bil05] Philip Bille. A survey on tree edit distance and related problems. *Theory Computer Science*, 337:217–239, 2005.

[BM07] Razvan C. Bunescu and Raymond J. 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] Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. Class-based ngram models for natural language. In *Computational Linguistics,18(4):468-479*, 1992.

[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.

[CBK+10] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI 2010)*, 2010.

[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 16th European Conference on Artificial Intelligence (ECAI 2004)*, 2004.

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

[CTB+01] Peter Clark, John Thompson, Ken Barker, Bruce Porter, Vinay Chaudhri, Andres Rodriguez, Jerome Thomere, Sunil Mishra, Yolanda Gil, Pat Hayes, and Thomas Reichherzer. Knowledge entry as the graphical assembly of components. In *K-CAP*, 2001.

[CV08] Sonia Chernova and Manuela Veloso. Teaching multi-robot coordination using demonstration of communication and state sharing. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2008.

[CW07] Philipp Cimiano and Johanna 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.

[DDF+90] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal Of The American Society for Information Science*, 41(6):391–407, 1990.

[DR06] Dmitry Davidov and Ari 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] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander 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.

[FL06] Alexander Faaborg and Henry Lieberman. A goal-oriented web browser. In *Proceedings of the 24th International Conference on Human Factors in Computing Systems (CHI 2006)*, 2006.

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

[GBM03] Roxana Girju, Adriana Badulescu, and Dan Moldovan. Learning semantic constraints for the automatic discovery of part-whole relations. In *Proceedings of the Human Language Technology Conference/Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL 2003)*, 2003.

[GBM06] Roxana Girju, Adriana Badulescu, and Dan 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] Zelig Harris. Distributional structure. In *Word, 10(23): 146-162s*, 1954.

[Hea92] Marti 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).* 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 in Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI 2008)*, 2008.

[HP82] M. D. Hendy and David Penny. *Branch and bound algorithms to determine minimal evolutionary trees.* 1982.

[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.

[KpKSS04] Karin Kailing, Hans peter Kriegel, Stefan Schnauer, and Thomas Seidl. Efficient similarity search for hierarchical data in large databases. In *Extending Database Technology*, pages 676–693, 2004.

[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.

[LC03] Dawn J. Lawrie and W. Bruce Croft. Generating hierarchical summaries for web searches. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2003)*, 2003.

[Lin74] Harold R. Lindman. *Analysis of variance in complex experimental designs*. W.H. Freeman & Co., 1974.

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

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

[Mac67] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the 5th Berkeley Symposium on Mathematical statistics and probability*, 1, 1967.

[Mah36] P. C. Mahalanobis. On the generalised distance in statistics. In *Proceedings of the National Institute of Sciences of India 2 (1): 495*, 1936.

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

[Mar07] Gary Marchionini. Beyond basic search. Presented in CS Colloquium, Computer Science Department, University of Illinois at Urbana-Champaign, 2007.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge UP, 2008.

[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 *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI 2005)*, 2005.

[NM03] Monica N. Nicolescu and Maja J. Mataric. Natural methods for robot task learning: instructive demonstrations, generalization and practice. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2003)*, 2003.

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

[PP06] Patrick Pantel and Marco 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] Patrick Pantel and Deepak Ravichandran. Automatically labeling semantic classes. In *Proceedings of the Human Language Technology Conference / Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL 2004)*, 2004.

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

[PTL93] Fernando Pereira, Naftali Tishby, and Lillian Lee. Distributional clustering of english words. In *Proceedings of the 31th Annual Meeting for the Association for Computational Linguistics (ACL 1993)*, 1993.

[RC98] Brian Roark and Eugene 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 (ACL/COLING 1998)*, 1998.

[RF07] Benjamin Rosenfeld and Ronen Feldman. Clustering for unsurpervised relation identification. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management (CIKM 2007)*, 2007.

[RH02] Deepak Ravichandran and Eduard 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.

[RhDM04] Erhard Rahm, Hong hai Do, and Sabine Mamann. Matching large xml schemas. In *Proceedings of the 23rd ACM SIGMOD International Conference on Management of Data (SIGMOD 2004)*, 2004.

[RM04] Paul M. Ramirez and Chris Mattmann. Ace: Improving search engines via automatic concept extraction. In *Proceedings of the 2004 IEEE International Conference on Information Reuse and Integration (IEEE IRI-2004)*, pages 229–234, 2004.

[RS97] Ellen Riloff and Jessica Shepherd. A corpus-based approach for building semantic lexicons. In *Proceedings of the 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 (ECAI 2004)*, 2004.

[SC99] Mark Sanderson and W. Bruce Croft. Deriving concept hierarchies from text. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1999)*, 1999.

[SC00] Greg Schohn and David Cohn. Less is more: active learning with support vector machines. In *Prococeedings of 17th International Conference on Machine Learning (ICML 2000)*, pages 839–846. Morgan Kaufmann, San Francisco, CA, 2000.

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

[SJN06] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (ACL/COLING 2006)*, 2006.

[SK01] Luc Steels and Frederic 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.

[STDC04] Idan Szpektor, Hristo Tanev, Ido Dagan, and Bonaventura Coppola. Scaling web-based acquisition of entailment relations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, 2004.

[TAJP07] Jaime Teevan, Eytan Adar, Rosie Jones, and Michael Potts. Information Re-Retrieval: Repeat Queries in Yahoo's Logs. In *Proceedings of the 30th Annual ACM Conference on Research and Development in Information Retrieval (SIGIR 2007)*, 2007.

[TB06] Andrea L. Thomaz and Cynthia Breazeal. Reinforcement learning with human teachers: evidence of feedback and guidance with implications for learning performance. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI 2006)*, 2006.

[TC09] Andrea L. Thomaz and Maya Cakmak. Learning about objects with human teachers. In *Proceedings of the 4th ACM/IEEE international conference on Human Robot Interaction (HRI 2009)*, 2009.

[TLBS03] Peter D. Turney, Michael L. Littman, Jeffrey Bigham, and Victor 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.

[TR06] Alicia Tribble and Carolyn Ros. Usable browser for ontologicial knowledge acquisition. In *Work-in-progress of the 24th International Conference on Human Factors in Computing Systems (CHI 2006)*, 2006.

[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 *Technical Report 208, Department of Statistics, Stanford University*, 2000.

[Voo01] Ellen. M. Voorhees. Evaluation by highly relevant documents. *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2001)*, 2001.

[Voo02] Ellen Voorhees. Overview of the trec 2002 question answering track. In *Proceedings of the Eleventh Text REtrieval Conference (TREC*, pages 115–123, 2002.

[War63] Joe H. Ward. Hierachical grouping to optimize an objective function. *Journal of the Ameirican Statistical Association*, 58:236–244, 1963.

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

[WMS02] Dennis Wackerly, William Mendenhall, and Richard L. Scheaffer. Mathematical statistics with applications. *Duxbury advanced Series*, 2002.

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

[Yan06] Liu Yang. Distance metric learning: A comprehensive survey. In *http://www.cs.cmu.edu/ liuy/frame_survey_v2.pdf*, 2006.

[YC06] Hui Yang and Jamie Callan. Near-duplicate detection by instance-level constrained clustering. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2006)*, 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 8th National Conference on Digital Government Research (Dg.O 2008)*, 2008.

[YC09a] Hui Yang and Jamie Callan. Feature selection for automatic taxonomy induction. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2009)*, 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.

[YCS06] Hui Yang, Jamie Callan, and Stuart Shulman. Next steps in near-duplicate detection for erulemaking. In *Proceedings of the 6th National Conference on Digital Government Research (Dg.O 2006)*, 2006.

[YKT05] Rui Yang, Panos Kalnis, and Anthony K. H. Tung. Similarity evaluation on tree-structured data. In *Proceedings of the 24th ACM SIGMOD International Conference on Management of Data (SIGMOD 2005)*, 2005.

[ZJ94] Kaizhong Zhang and Tao Jiang. Some max snp-hard results concerning unordered labeled trees. In *Information Processing Letters, 49:249-254.*, 1994.

[ZKB02] Jrgen Ziegler, Christoph Kunz, and Veit Botsch. Matrix browser - visualizing and exploring large networked information spaces. In *Proceedings of the 22nd International Conference on Human Factors in Computing Systems (CHI 2002)*, 2002.

[ZL04] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22:179–214, April 2004.

[ZS89] Kaizhong Zhang and Dennis Shasha. Simple fast algorithms for the editing distance between trees and related problems. *Society for Industrial and Applied Mathematics (SIAM) Journal on Computing (SICOMP)*, 18(6):1245–1262, 1989.

[ZSS92] Kaizhong Zhang, Rick Statman, and Dennis Shasha. On the editing distance between unordered labeled trees. *Information Processing Letter*, 42(3):133–139, 1992.

[ZZSW09] Huibin Zhang, Mingjie Zhu, Shuming Shi, and Ji-Rong Wen. Employing topic models for pattern-based ssemantic class discovery. In *Proceedings of the 47th Annual Meeting for the Association for Computational Linguistics (ACL 2009)*, 2009.