Browsing Hierarchy Construction by Minimum Evolution

Hui Yang, Department of Computer Science, Georgetown University

Hierarchies serve as browsing tools to access information in document collections. This paper explores techniques to derive browsing hierarchies that can be used as an information map for task-based search. It proposes a novel minimum-evolution hierarchy construction framework, which directly learns semantic distances from training data and from users to construct hierarchies. The aim is to produce globally optimized hierarchical structures by incorporating user-generated task specifications into the general learning framework. Both an automatic version of the framework and an interactive version are presented. A comparison with state-of-the-art systems and a user study jointly demonstrate that the proposed framework is highly effective.

Additional Key Words and Phrases: Browsing hierarchy construction, Document organization

1. INTRODUCTION

As Web users have become more involved in Web search, search tasks have become more complex and dynamic. However, examining a long list of documents multiple times to accomplish a task is still a common experience for users. It places too much of a burden on the user to make decisions when the task complexity is great. It suggests that the overhead of such manual efforts should be minimized. It also indicates that existing one-shot keyword search is inadequate for complex search. Recent developments in information retrieval (IR) have attracted attention to complex search in a bigger context. The efforts include TREC Contextual Suggestion Track [Dean-Hall et al. 2012], TREC Session Track [Kanoulas et al. 2013], dynamic IR models such as the POMDP models [Luo et al. 2014; Zhang et al. 2013], online learning models [Hofmann et al. 2011; Hofmann et al. 2013] economic models [Azzopardi 2014], and many user-centric research efforts in complex search [Lagun and Agichtein 2011; Kelly et al. 2010].

What is missing in supporting complex and task-based search is to provide a clear picture of the context, to assist users during a search task. We are motivated to create a tool, such as an "information map," that is able to show the users where they are in the "information space," and what the current context of the search is. We propose to build such a tool and allow users to explore it and browse the documents. Hierarchies are proposed here to organize the documents, as hierarchical organization is the most commonly used structure for organization.

In fact, organizing documents is a common task in information seeking and knowledge management. Metasearch engines such as Clusty (now Yippy) employ search result clustering to automatically organize search result documents into browsing hierarchies [Guan and Yang 2013; Luo et al. 2013]. The algorithms we develop here are able to support browsing for search with or without a query. This paper studies how to build a browsing hierarchy automatically from a set of documents, including search results. We also study how to incorporate light-weight manual edits to improve the quality of the browsing hierarchy.

We view a browsing hierarchy and the document collection that it supports as a whole.[1] This entire entity consists of three components – the documents, the nodes with labels in the hierarchy, and the hierarchy that provides the organization. The browsing hierarchy can be considered an ontology that shows an overview of the document set. The nodes in this hierarchy are linked back to the documents. Users can thus browse in this hierarchy and explore the document set.

Clustering has been a popular approach for browsing hierarchy construction. It seems suitable for this task, and often becomes the choice to start with. The common steps for clustering approaches are (1) first clustering documents, and then (2) assigning labels to the clusters [Cutting et al. 1992; Ke et al. 2009; Carpineto and Romano 2010].

[1]The following terminology is used in this paper. *Collection* refers to a document collection, including a set of search results. *Browsing hierarchy* or *hierarchy* refers to the tree-structured hierarchical organization of topics in a collection. A *term* refers to a noun or a noun phrase that could potentially be a node.

We argue that clustering algorithms may not be the ideal solution for browsing hierarchy construction, for the following reasons:

— Clustering often produces clusters with mixed initiatives. For instance, items can be organized in different ways, and many different topics may be mixed in one cluster. This makes the clustering results difficult to interpret. This problem derives from the polythetic nature of the features that clustering relies on to build the hierarchy.
— Clustering labels are expected to present the main topic of a cluster; however, labels produced by automated algorithms are often not good enough to describe the clusters.

Due to clustering approaches' limitations, we are motivated to look for alternative solutions. Our algorithm is inspired by how a human builds hierarchies for a set of items. Card sorting is a common technique used for this task [Chaparro et al. 2008]. It is an incremental organizing process: The items are examined one by one or a few at a time; then, based on those in hand, we build a small hierarchy or hierarchy fragments of the items. We then examine more items and add them into the hierarchies that we just built. When adding new items into the hierarchy, we make sure that they are placed in the most suitable places. The criteria that define the suitableness are usually task-specific, and could change as the user is constructing the hierarchy.

Following this idea, in this paper, we propose a novel hierarchy construction framework. We first perform node extraction to acquire nodes from a given dataset. ("Nodes" are topics of interest in the given dataset. They are usually nouns, noun phrases, or named entities. They are directly extracted from the document collection.) The proposed algorithm is not constrained to a given type of collection. It can be applied either to an entire document collection, or to just a number of documents that show up in search results. However, the size of the collection does affect the speed at which the hierarchy can be constructed.

After the nodes are extracted, the next step is to organize them. We employ a fully automated process, which we call "minimum evolution," to create a browsing hierarchy. The framework incrementally clusters nodes based on semantic distances between nodes, and transforms the task of browsing hierarchy construction into a multi-criterion optimization process, based on optimization of the hierarchy structure. The distances are learned from the training data and from the user who constructs the hierarchy. A wide range of features, including lexico-syntactic patterns, context, and definitions from dictionaries, are used to derive distances between nodes.

We also present an interactive process that adopts a human-guided machine learning approach. In this interactive process, the user interacts with the system and makes improvements to the hierarchy. The system learns what the user changes, and adapts to make sensible predictions about how to group the remaining unorganized nodes. After receiving the system's suggestions, the user evaluates them and makes more improvements if necessary. Through several iterations of interaction between the user and the system, a browsing hierarchy that satisfies the user's needs is finally constructed. A comparison to state-of-the-art systems and a user study jointly demonstrate that the proposed framework is highly effective.

The proposed framework makes the following contributions in the field of browsing hierarchy construction:

— We put our focus on the browsing hierarchy's accuracy, which contributes to building an organization of the documents with a hierarchy that 'look correct.' Our way of constructing the hierarchy leads to a novel procedure which produces the most 'correct' hierarchy of those produced by existing approaches.
— We propose a novel unified optimization framework that is able to flexibly incorporate a wide range of heterogeneous features from patterns and from statistical measures, such as co-occurrence and contextual distributions. It is the first piece of work that seamlessly integrates features from both families of statistical measures and linguistic patterns.

— We also propose a novel online learning algorithm that collects training data from the user, through which we incorporate personalization and task specifications seamlessly in an interactive setting. Users are able to build hierarchies in their own ways by interacting with the system.

The majority of our work has been devoted to organizing the nodes, with a focus on putting them into the right positions – a process we think is missing in most data-driven approaches. Using clustering as an example, we often find that many labeled clusters in the hierarchy do not seem 'correct.' It could be that they have the wrong labels for their clusters, that they are misplaced, that they should be further grouped or further split, etc. It can be difficult to define what is 'correct.' Our approach takes this into account and incorporates constraints to ensure the hierarchy 'looks correct.' With some concessions to scalability, we achieve a high user-perceived accuracy in the user study. Our approach can be particularly suitable for small-scale collections, such as search results, where small but 'correct' hierarchies are desirable.

The remainder of this paper is organized as follows. Section 2 describes related work. Section 3 explains how to obtain nodes from a collection. Section 4 details our formulation of semantic distances between nodes, and the features that we used. Section 5 presents the novel minimum-evolution framework for automatic hierarchy construction. Section 6 describes how to incorporate constraints in the framework to handle path inconsistency. Section 7 details the interactive browsing hierarchy construction framework. Section 8 describes our evaluation of our results. Section 9 contains a discussion of our work, and Section 10 concludes the paper.

2. RELATED WORK

Prior research on browsing hierarchy construction can be classified into four main classes: *clustering-based* approaches, *monothetic concept hierarchy* approaches, *linguistic-based* approaches, and *facet construction* approaches.

The common steps for *clustering-based* approaches are first clustering documents and then assigning labels to the clusters [Cutting et al. 1992; Ke et al. 2009; Carpineto and Romano 2010]. They follow a sequence of "cluster then label." For example, Scatter/Gather *scatters* a document collection into clusters and provides labels and summaries of the clusters [Cutting et al. 1992]. The user selects clusters that most interest him/her; the algorithm *gathers* the new selections from the user to form a smaller document sub-collection and *scatters* it again into sub-clusters. The process repeats until the new sub-collection is small enough to only hold one or a few documents. Each cluster is labeled by multiple terms that appear frequently in the cluster.

Clustering-based approaches often produce non-interpretable clusters, due to their data-driven nature and poor cluster labeling. Even in the best-known commercial clustering-based search engine, Clusty (now Yippy[2]), which presents search results in hierarchical clusters and labels the clusters by variable-length sentences, cluster labeling remains a challenge.

Several clustering-based approaches have studied personalization. For instance, Scatter/Gather invites users to provide feedback to *change the document space*. They incorporate personalization by gathering users' preferred documents into a sub-collection to reach the relevant documents that eventually satisfy their information needs. Other clustering approaches allow users to *tweak the features or constraints* in clusters. For example, Kerne et al. clusters documents based on meta-data such as document location [Kerne et al. 2005]. It allows the user to participate in feature selection and interact with the clustering algorithm by setting relative weights for the features. Huang and Mitchell studied mixed-initiative clustering in Gaussian mixture models [Huang and Mitchell 2007]. The algorithm presents the users with properties such as labels for instances, must-links among instances, cluster existence, and key terms in clusters. A user can provide approval, disapproval, modification, or new properties to the algorithm; the algorithm re-trains the model by adjusting the probabilities of the properties, based on the feedback.

[2]http://www.yippy.com

In the interactive version of our algorithm, we also collect user feedback during browsing hierarchy construction. Our focus is on learning a user-defined distance metric that indicates how they organize the hierarchy. The algorithm applies the newly learned distance metric function to unclustered nodes, so that the browsing hierarchy is personalized.

Monothetic methods have been proposed as an alternative solution to polythetic clustering algorithms. Subsumption [Sanderson and Croft 1999] is a representative *monothetic concept hierarchy construction* method. Through describing and exploring terms' document frequency, subsumption decides the parent-to-child relationship between nodes in a hierarchy based on conditional probability, a monothetic feature. The *monothetic concept hierarchy* approaches avoid clustering labeling by first extracting concepts from documents, and then organizing concepts into a hierarchy where each node attaches to the subset of documents containing it [Sanderson and Croft 1999; Lawrie et al. 2001; Kummamuru et al. 2004]. It follows a sequence of "label then cluster." Lawrie et al. [Lawrie et al. 2001] proposed a level-by-level top-down hierarchy construction scheme to discover topic words as concepts. At each level of a hierarchy, they identify concepts that are heavily connected to all other words. Adeyanju et al. [Adeyanju et al. 2012] utilized subsumption to build hierarchies from both the collection and search logs for Web search query recommendation. They proved that the monothetic approach can work on both large and small datasets, such as intranet search results.

It is worth noting that the "monothetic" feature – conditional probability – is not a feature that describes a node; it describes a relation between two nodes. Moreover, subsumption avoids the issues related to clustering labeling by extracting nodes from documents and organizing them based on conditional probabilities. Here a node is considered as a coherent concept that is usually a noun phrase. It can also be considered as a label for a cluster. *Monothetic concept hierarchy* approaches could produce hierarchies that lack semantic meanings, because no semantic feature is taken into account.

In contrast to clustering-based and monothetic approaches, *linguistic-based* or *pattern-based* approaches are able to produce hierarchies that look more 'correct,' because they often use natural-language-understanding techniques to derive relations among nodes [Hearst 1992; Yang and Callan 2009; Crouch 1988; Etzioni et al. 2005]. These approaches are mostly proposed by the natural language processing (NLP) communities for automatic taxonomy construction. Lexico-syntactic patterns are widely used to discover relations either by exhaustive search or by bootstrapping. One of their well-known weaknesses is their poor coverage, due to the paucity of natural language patterns.

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 [Berland and Charniak 1999; Kozareva et al. 2008] or via automatic bootstrapping [Hearst 1992; Etzioni et al. 2005; Girju et al. 2003; Ravichandran and Hovy 2002; Pantel and Pennacchiotti 2006]. Traditional research on taxonomy construction focuses on extracting local relations between node pairs [Hearst 1992; Berland and Charniak 1999; Ravichandran and Hovy 2002; Girju et al. 2003; Etzioni et al. 2005; Pantel and Pennacchiotti 2006; Kozareva et al. 2008]. Recent efforts have involved building full taxonomies. For example, [Snow et al. 2006] proposed to estimate taxonomic structure via maximizing the overall likelihood of a taxonomy. [Kozareva and Hovy 2010] proposed to connect local node pairs by finding the longest path in a subsumption graph. 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). The first noun phrase *NPy* indicates the parent, the second noun phrase *NPx* indicates a child, and the third noun phrase *NPz* indicates another child. This pattern is used to find instances for *is-a* relations, particularly to find more children for a known parent. The double-anchored pattern is more specific, since both the parent and a child are required to appear in the pattern. Not surprisingly, the double-anchored pattern sacrifices coverage. The pattern can also bootstrap to get more instances, and then substitute the new instances as the child in another double-anchored pattern. Bootstrapping increases coverage while introducing many incorrect instances. In our work, we use highly accurate patterns as features in our learning framework. However, we do not rely only on them. We also include statistical features, to achieve a balance between coverage and accuracy.

As bottom-up approaches, both subsumption and pattern-based methods can suffer from the problem of inconsistent long-distance paths. Path inconsistency, also known as node intransitivity, is a common problem in many approaches that build hierarchies locally from pair-wise nodes. For example, a hierarchy built by subsumption or patterns could consist of a path *financial institute→bank→river bank*, which is inconsistent because *financial institute* should not be ascendant of *river bank* even if the immediate relations are true. Adding transitivity to hierarchies constructed by subsumptions has been studied in [Sanderson and Lawrie 2000]. In this paper, we propose global optimization over the entire hierarchy to deal with this issue.

The *facet construction* approach mainly serves for faceted search [Yee et al. 2003]. Researchers usually apply supervised learning to classify nodes into existing categories, or carve out browsing hierarchies from existing ontologies. For example, Stoica et al. [Stoica and Hearst 2007] managed to extract a browsing hierarchy from hypernym relations within WordNet [Fellbaum 1998]. Dakka et al. classified concepts into pre-defined categories to build facets [Dakka et al. 2005]. In this paper we focus on building browsing hierarchies to assist complex and task-based search. Hierarchies using existing ontologies or portions of them cannot easily adapt to specific tasks. Thus this approach is not suitable for our task and will not be compared in the evaluation section.

In addition, a practical personalized clustering algorithm was proposed by [Wang and Zhai 2007] for search result organization. They used search logs to help build the browsing hierarchy. First, possible related categories for a search topic are identified using the search logs, and are used as the nodes in the hierarchy. Second, the hierarchy is formed by a clustering algorithm; the personalization of the hierarchy is done through using the nodes as centroids. Third, the documents are categorized into the nodes in the hierarchy.

In this paper, we first extract nodes, then organize them to form browsing hierarchies. We then classify the documents into their most relevant nodes by using the Support Vector Machine (SVM) classifier.[3] This is similar to the *monothetic* approaches and [Wang and Zhai 2007]. We employ linguistic features to derive hierarchies; this is similar to the *linguistic* approaches. However, our approach is distinguished from other approaches in that it globally optimizes the browsing hierarchy's overall structure and provides a general learning framework to incorporate heterogeneous features and user feedback.

3. NODE EXTRACTION

The first step in our framework is to extract a set of nodes that can represent the topics in documents.

Instead of being selective about the final set of nodes that eventually appear in the hierarchy, we exhaustively examine all terms in the document collection, and keep a good portion of them as candidates for later use. The main steps of node extraction are:

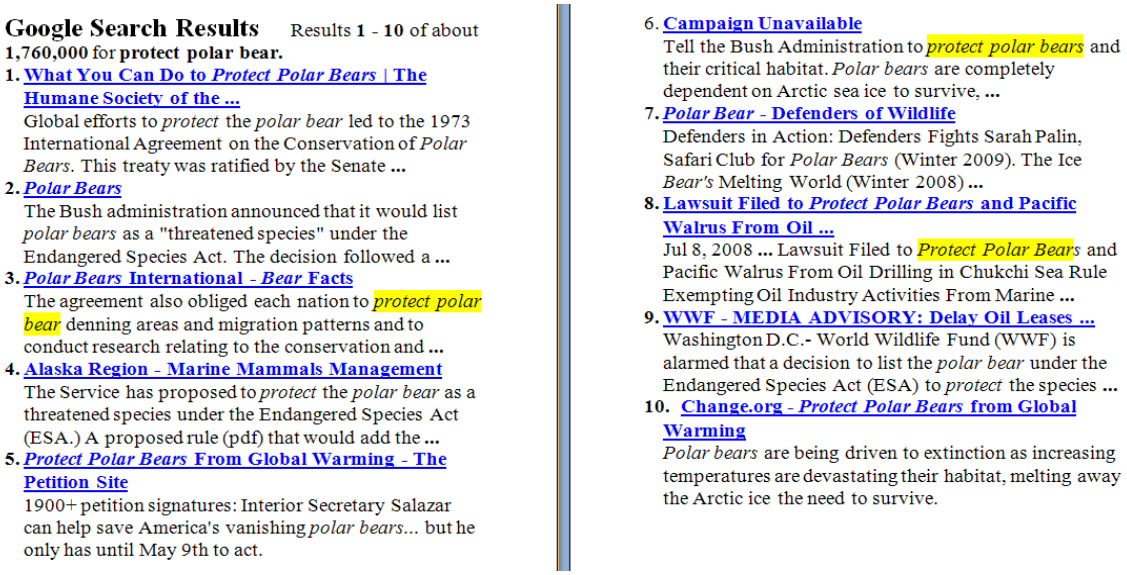
(1) Node mining. This process exhaustively examines the collection and selects nouns, noun phrases[4], and named entities[5] that occur >5 times in the collection as node candidates. Node mining tries to extract *all* possible node candidates from a document collection. We consider nodes to be nouns or noun phrases. We therefore extract nominal unigrams, bigrams, trigrams, and even longer noun sequences as node candidates. Documents are split into sentences before we perform the extraction. Each sentence is labeled by a part-of-speech (POS) tagger,[6] which annotates each word with its linguistic part-of-speech. 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).

[3]http://svmlight.joachims.org/

[4]We use Stanford NLP Parser to get the part-of-speech (POS) tags.

[5]We use BBN Identifier to obtain the named entities.

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



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.

Fig. 1. Google snippets for *protect polar bear* (an error): three occurrences

(2) Node filtering. Node mining produces many node candidates. There are false positives among them–for instance, "protect polar bear," which is not a noun phrase. Many errors are caused by POS tagging. We observe that noun phrases are used frequently in our daily language. However, the occurrence of a verb or an adjective preceding a noun phrase is much less frequent. That is to say, the frequency of "protect polar bear" is much less than that of "polar bear." Therefore, we design a Web-based frequency test to filter out false positives. We submit each candidate to *google.com* as a search query, then remove invalid terms due to part-of-speech errors or misspelling, by removing terms that occur <4 times within the top ten returned snippets. Query terms that appear in the title are not used when counting occurrences because they have a high probability of overlapping with the query terms. We therefore use only snippets for this Web-based filtering. Figure 1 shows the snapshot of the first ten Google snippets for *protect polar bear*. Since there are only three occurrences of this phrase (< 4), the Web-based frequency test determines that it is not a valid node candidate. If there are more than four occurrences of a phrase (a threshold that is empirically set), we judge it as a valid node candidate. The method is quite robust, so the probability of missing true positives is low. For instance, another three-word phrase, "nuclear power plant," shows up >10 times in its first ten Google snippets, which makes it a good candidate, and we can recognize it.

(3) Node unification. We conflate similar nodes into one. We first cluster terms by LSA (latent semantic analysis) [Bellegarda et al. 1996] and select the most frequent terms as nodes from each term group. The procedure is the following.
   (a) Create a term-document matrix C. Each entry C_{ij} in C represents the tf.idf weight of term c_i in document d_j.
   (b) Perform singular value decomposition (SVD) for C and obtain the SVD term matrix U, singular value matrix Σ, and SVD document matrix V. The rank of C is $rank$, which indicates the first $\mathcal{R}$ non-zero diagonal entries in Σ.
   (c) Reduce the rank of C from $\mathcal{R}$ to $\mathcal{K}$, where $\mathcal{K}$ is an integer much smaller than $\mathcal{R}$. We empirically test different $\mathcal{K}$ values and choose the $\mathcal{K}$ value based on [Tibshirani et al. 2000]. Basically, $\mathcal{K}$ is selected if it generates an "elbow value" in the summary purity scores for

the series of node clustering results. An "elbow value" is a dramatic change of the slope that a line or a plot shows. $\mathcal{K}$ is tested from 50 to 500 in our experiments.

(d) Given a $\mathcal{K}$, set the last $(\mathcal{R} - \mathcal{K})$ diagonal entries in Σ to zeros to obtain $\Sigma_{\mathcal{K}}$. Calculate a truncated node-document matrix at rank-$\mathcal{K}$: $C_{\mathcal{K}} = U\Sigma_{\mathcal{K}}V^T$.

(e) Compute the term-term matrix $C_{\mathcal{K}}C_{\mathcal{K}}^T$, whose $(m, n)^{th}$ entries indicates the similarity between two terms c_m and c_n in the lower dimensional space. Cluster the terms based on the scores in $C_{\mathcal{K}}C_{\mathcal{K}}^T$. Repeat steps (c), (d) and (e) to pick the best $\mathcal{K}$.

(f) From each cluster, choose a term with the highest corpus frequency to keep in the node set C.

(4) Node trimming. Depending on the size of the document collection, the number of nodes that we have after node unification could still be large. In this work, we trim down the size of the node sets by first sorting the remaining nodes by their document frequency in descending order, and then just keeping the first n nodes in the sorted list and putting them into the node set C. n is a parameter that the user can set. It corresponds to the size of the hierarchy that the user would like to build.

At the end of the node extraction, a set of ranked nodes are generated. The nodes are noun phrases that our algorithm considers to best represent the document set. The nodes are ranked by their inverse document frequency, in decreasing order. As we mentioned earlier, the size of the node set will be the hierarchy size. The size n is a user-set parameter. In our experiments, n ranges from 40 to 200.

4. MODELING SEMANTIC DISTANCE BETWEEN NODES

Our framework then organizes the nodes into a hierarchy. The organizing process relies on pairwise distances calculated for nodes. We call this distance "semantic distance." Our goal is to find a parametric distance metric function to represent the "semantic distance," and to allow combining a wide range of heterogeneous features and assigning them different weights. It also must produce distances that satisfy the requirements of a valid metric. The semantic distance we defined here can either be symmetric or asymmetric. When it is used to formulate symmetric relations, such as sibling relations, the distance is symmetric. When it is used to formulate asymmetric relations, such as parent-to-child relations, the distance is asymmetric.

We propose to model the semantic distance between nodes (c_x, c_y) as a Mahalanobis distance [Mahalanobis 1936]. Mahalanobis distance is a general parametric function widely used in distance metric learning. It measures the dissimilarity between two random vectors of the same distribution with a covariance matrix W. When only diagonal values of W are taken into account, W is equivalent to assigning weights to different features.

Specifically, a Mahalanobis-formatted semantic distance for nodes c_x and c_y can be represented as:

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

where $\Phi(c_x, c_y)$ represents the set of pairwise underlying feature functions, each feature function is $\phi_k : (c_x, c_y)$ with k=1,...,$|\Phi|$. W is the covariance matrix, whose diagonal values weigh the underlying feature functions. We can estimate W by minimizing the squared errors between the semantic distances generated from training data and the expected distances.

To be a valid metric, a semantic distance d has to fulfill the following criteria:

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

These requirements are used in our framework as constraints for learning valid semantic distances. The equality condition can be trivially satisfied. We therefore focus on other requirements. When W is properly constrained to be positive semi-definite (PSD) [Bhatia 2006], denoted as $W \succeq 0$, a

Table I. Lists of lexico-syntactic patterns. In a hypernym pattern, NP_x indicates the slot for a hyponym node and NP_y indicates the slot for the hypernym node. In a part-of pattern, NP_x indicates the slot for a meronym node and NP_y indicates the slot for the coordinate node. In a sibling pattern, both NP_x and NP_y indicate the slots for two sibling nodes. ? indicates optionality.

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		

Mahalanobis-formatted distance will be guaranteed to satisfy non-negativity and triangle inequality. We thus constrain W to be PSD ($W \succeq 0$).

Provided D as the training node set and d_{c_x,c_y} as a training distance, the parameter estimation for W can be achieved by optimizing the following objective:

$$\min_{W} \sum_{x=1}^{|\mathbf{D}|} \sum_{y=1}^{|\mathbf{D}|} \left(d_{c_x,c_y} - \sqrt{\Phi(c_x,c_y)^T W^{-1} \Phi(c_x,c_y)} \right)^2, \text{ subject to } W \succeq 0 \tag{2}$$

The optimization is done by using standard semi-definite programming (SDP) solvers. We use MATLAB packages SeDuMi[7] and YALMIP[8] to perform the optimization in our implementation.

In our framework, a major source of training data is existing ontologies such as WordNet [Fellbaum 1998] and ODP (the Open Directory Project) [ODP 2011]. We obtain the semantic distance for a node pair (c_x, c_y) in training data by summing up edge weights along the shortest path from c_x to c_y in a training hierarchy. The edge weight can be assigned based on the types of relations that an edge represents. For example, if the relation r is *is-a*, in a training distance matrix, the parent-child pairs are indicated as 0, and the other nodes are indicated as 1. If the relation r is a *sibling* relation, such as 'apple' and 'pear,' their within-cluster distances are defined as 0 and their between-cluster distances are defined as 1. The learned model W is used to predict distance scores for testing node pairs by applying Eq. 1 to them.

We assume that there exist some underlying feature functions that can measure semantic dissimilarity between two nodes from various aspects, and a good semantic distance involves all these features. Thus, obtaining good features is crucial. We report below the effective features that we discovered.

4.1. Features

We employ a wide range of features to cover various aspects of measuring distances between nodes. Given two nodes c_x and c_y, a feature is defined as a function $\phi : (c_x, c_y)$ that generates a value within [0,1]. In total, we design more than 30 features, including features using *lexical-syntactic patterns, context, co-occurrence, syntactic dependency*, and *definitions*. We describe each type of feature below.

4.1.1. Lexico-Syntactic Patterns. Table I lists all 17 lexico-syntactic patterns used in this research. The pattern-based features include hypernym patterns [Hearst 1992; Snow et al. 2005], sibling patterns, and part-of patterns [Girju et al. 2003; Cimiano and Wenderoth]. For a hypernym pattern, NP_x indicates the slot for a hyponym node and NP_y indicates the slot for a hypernym node. For a part-of pattern, NP_x indicates the slot for a meronym node and NP_y indicates the slot for a coor-

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

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

dinate node. For a sibling pattern, both NP_x and NP_y indicate the slots for two sibling nodes. The feature function returns a vector of scores for two nodes, one score per pattern. A score is 1 if two nodes match a pattern in the document collection, and 0 otherwise.

4.1.2. Contextual Features. As stated in the Distributional Hypothesis [Harris 1970], words appearing in similar contexts tend to be similar. Therefore, word meanings can be inferred from and represented by contexts. Hence, we develop the following contextual features, including one global context feature and five local context features.

Global Context KL Divergence. This feature function measures the Kullback-Leibler divergence (KL divergence) [Strohman et al. 2005] between two unigram language models associated with two input nodes c_x and c_y. The global context of a node is considered as the documents containing it. We build the global context for each node into a unigram language model. The KL divergence between two language models associated with the two input nodes is used as the output value for this feature function. In particular, if the language model for node c_x is l_x, and for c_y is l_y, the KL divergence [Croft et al. 2004] can be calculated as:

$$D_{KL}(l_x||l_y) = \sum_i x_i log_2 \frac{x_i}{y_i}, \tag{3}$$

where i is the index for the i^{th} word in l_x, x_i is this word's frequency in l_x, and y_i is its frequency in l_y. If $x_i = 0$ or $y_i = 0$, Dirichlet smoothing [Zhai and Lafferty 2004] is applied.

Local Context KL Divergence. We extract local contexts for nodes and compare their local contexts from the corpus. We define the local context of a node c_x as the left δ and the right δ words surrounding it. δ is a sliding window size that ranges from 1 to 5 in our settings (these numbers are empirically set). Among all the documents containing c_x in the collection, the words surrounding c_x are collected and built into a unigram language model for the node. Similarly, we derive a unigram language model for the other node c_y using its surrounding words, and output the KL divergence of the two unigram models as the score. We have five features generated from this function with different window sizes δ.

4.1.3. Co-occurrence Features. We measure the co-occurrence of two nodes c_x and c_y by their point-wise mutual information (PMI):

$$pmi(c_x, c_y) = log \frac{p(c_x, c_y)}{p(c_x) \times p(c_y)}, \tag{4}$$

where the the probability function p is calculated by $Count()$/corpus size. Since the corpus size is a constant, we can calculate the PMI as $pmi(c_x, c_y) = log \frac{Count(c_x, c_y)}{Count(c_x) \times Count(c_y)}$, where the $Count()$ function can be defined at different levels of granularity: it can be the number of *documents* containing one or both of the nodes, or the number of *sentences* containing c_x or c_y or both. Based on different definitions for the $Count()$ function, we have the following three co-occurrence features: *document-level PMI*, where $Count()$ is defined as the number of documents in a dataset containing the node(s); *Sentence-level PMI*, where $Count()$ is defined as the number of sentences in a dataset containing the node(s); and *Google PMI*, where $Count()$ is defined as n for node 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 node pair, we query google.com with a concatenation of the two nodes.

4.1.4. Minipar Syntactic Distance. This feature function measures the number of edges between two nodes c_x and c_y in a syntactic parse tree. For each sentence s_i that contains both c_x and c_y, we use Minipar[9] to generate a syntactic parse tree t_{s_i} for s_i; c_x and c_y appear as two nodes in t_{s_i}. Let x_i be the shortest edge distance – i.e., the number of edges – between c_x and c_y in t_{s_i}. This feature function outputs an averaged value of x_i for all i where sentences s_i contain both c_x and c_y.

[9]http://webdocs.cs.ualberta.ca/~lindek/minipar/.

4.1.5. Modifier Overlap. This feature function measures the number of overlaps between modifiers for each of the two nodes c_x and c_y. Suppose there is a sentence s_i that contains c_x. Within the Minipar parse tree for s_i, we define a *modifier* to a node 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 node c_x is "federal rule," a parse 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 modifiers for "federal rule." Thus the modifier in this sentence for "federal rule" is "EPA proposed." Similarly, we can find the modifiers for c_y. We gather all unique modifiers for c_x and c_y in all sentences in the document collection. The number of overlaps between the two lists of modifiers is the score for this feature and can be calculated as: $|modifier(c_x) \bigcap modifier(c_y)|$.

4.1.6. Object Overlap. This feature function measures the number of overlaps between objects in sentences with two nodes, c_x and c_y, as the sentences' subjects. The objects and other semantic roles (such as subjects or verbs) are recognized by a semantic role tagger ASSERT[10]. Suppose a sentence s_i contains c_x. We first parse the sentence with ASSERT. The object for a node 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." The object is "power plants," which is labeled as "ARG1." 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 document collection. 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 nodes as the output value for this feature function.

4.1.7. 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 nodes 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") and the subjects are labeled as SUBJ (or "ARG0"). We gather all the unique objects for c_x in all sentences in the document collection.

4.1.8. 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 nodes in a similar manner. The only difference is that c_x and c_y are either labeled as SUBJ (or "ARG0") or OBJ (or "ARG1") 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.

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

4.1.10. Definition Overlap. We also measure how similar the definitions for two nodes c_x and c_y are. In particular, we submit a query for c_x to google.com or WordNet [Fellbaum 1998] 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.com. Google returns a page containing

[10]http://cemantix.org/software/assert.html

a list of entries, each explaining the meaning of the word. Here we only show three definitions for "government":

— the governing body of a nation, state, or community - "an agency of the federal government" [10/2/2013]
— 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 stopwords, and applying stemming, we compare the number of word overlaps between the definitions for the two nodes and normalize the value by the length of the shorter definition list.

5. HIERARCHY CONSTRUCTION BY MINIMUM EVOLUTION

With the nodes being extracted and the pair-wise semantic distance between nodes being learned, we are ready to construct the hierarchy. This section presents a novel automatic hierarchy construction framework termed "minimum evolution." The name *minimum evolution* comes from the minimum evolution theory in biology and the minimum evolution tree selection criterion widely used in phylogeny (Hendy and Penny, 1985).

We are inspired by our observation of how people sort items into hierarchies. Card sorting is a common technique that people use [Chaparro et al. 2008]. It is an incremental organizing process: The items are examined one by one or a few at a time; then, based on those in hand, a small hierarchy or hierarchy fragments of the items are built. We then examine more items and add them into the hierarchies that we just built. When adding new items into the hierarchy, we usually make sure that they are placed in the most suitable places. The criteria that define the suitableness are usually task-specific, and could change as the user is constructing the hierarchy.

We can sort the distance between all node pairs in advance, and then add nodes to their nearest neighbors. However, this strategy tends to produce hierarchies with long chains, which is not desirable. We therefore cannot simply use the nearest neighbors as the best positions at which to add new nodes. We thus design our framework as one that incrementally builds a browsing hierarchy by considering the nodes one at a time and inserting each at an optimal position within the hierarchy.

Our framework approaches the problem of hierarchy construction with two important assumptions:

(1) First, we assume that a good browsing hierarchy should reflect the proximity among nodes. This suggests that positioning a node at its correct position is to put it in a neighborhood with correct parents, children, and siblings. Its semantic distance to its true neighbors should be small, and to its false neighbors, big. Therefore, an optimal browsing hierarchy will produce minimum semantic distances among the nodes. This desirable property produces a guideline for us to select the best hierarchical organization. For the sake of efficiency, we can use a summary statistic, the sum of the semantic distances for all nodes, as a single measure for one hierarchy. That is to say, a good hierarchy minimizes the overall semantic distances summed over all node pairs.
(2) Second, every time the hierarchy grows, dramatic changes in the hierarchy should be avoided. Therefore, the hierarchy only "evolves minimally." After adding a new node, the new hierarchy should represent a minimal structural change from the previous one. The best next hierarchy is the one that introduces the least change in the overall semantic distances. This assumption can be considered as a greedy version of the first assumption.

To provide notation for the minimum evolution framework, we first introduce a few concepts. Given the node set C, we define a *full hierarchy* as a tree to which we attempted to add all nodes discovered by techniques presented in Section 3:

$$T_{full} = (C, R)$$
$$s.t.\ \forall c_x \in C, c_y \in C, c_x \neq c_y, \exists r(c_x, c_y) \in R, \tag{5}$$

where C is the node set and R is the relation set defined over $C \times C$.

A full hierarchy is constructed by adding nodes one at a time, which yields a series of partial hierarchies T^0, T^1, T^2, ... , and T^n, where n is the number of nodes in C. A *partial hierarchy* is a tree containing only a subset of nodes in C:

$$T_{partial} = (C, R)$$
$$s.t.\ \exists c_x \in C, c_y \in C, c_x \neq c_y, r(c_x, c_y) \notin R, \tag{6}$$

where C is the node set and R is the relation set defined over $C \times C$.

We define an *information function* as the sum of the semantic distances for a hierarchy. The information function is a measure of information represented by a hierarchy or part of a hierarchy. We name it this way because the sum of semantic distances can be viewed as the amount of information or surprises in a hierarchy. For a hierarchy $T(C, R)$, its *information function* $Info(T)$ is the sum of the pairwise distances among all nodes in T:

$$Info(T) = \sum_{c_x, c_y \in T} d(c_x, c_y), \tag{7}$$

where x, y are the indices for the two nodes c_x and c_y.

Following the first assumption, we can define that the goal of browsing hierarchy construction is to find an optimal full browsing hierarchy $\hat{T}$ such that the overall semantic distance is minimized; i.e., to find:

$$\hat{T} = \arg\min_{T'} Info(T') \tag{8}$$

.

The second assumption – which is the greedy version of the first assumption – suggests that when adding a new node, say the n^{th} node, the best current browsing hierarchy $\hat{T^n}$ is one that introduces the least change in overall semantic distances from the previous browsing hierarchy T^{n-1}:

$$\hat{T^n} = \arg\min_{T'^n} \Delta Info(T'^n, T^{n-1}) \tag{9}$$

Since the node set for a full hierarchy is always C, the only unknown for $\hat{T^n}$ is the relations $\hat{R^n}$. Thus, since we only need to optimize over R, Eq. 9 can be transformed into:

$$\hat{R^n} = \arg\min_{R'^n} \Delta Info(T'^n(S^n, R'^n), T^{n-1}(S^{n-1}, R^{n-1})), \tag{10}$$

where S^n is the n^{th} node set, R^n is the n^{th} relation set.

By plugging in the definition of semantic distance and information function, the updating function becomes the minimization of the absolute difference in the hierarchy's overall semantic distance with and without the n^{th} new node c_z:

$$R^{(n)} = \arg\min_{R'} | \sum_{c_x, c_y \in S^{n-1} \cup \{c_z\}} d(c_x, c_y) - \sum_{c_x, c_y \in S^{n-1}} d(c_x, c_y)|, \tag{11}$$

where $d_{(c_x, c_y)}$ is the semantic distance between c_x and c_y.

Algorithm 1 outlines a greedy minimum evolution algorithm for hierarchy construction. The algorithm inserts each node into the hierarchy so that the insertion is equivalent to minimizing the overall semantic distances. The algorithm searches through all the possible positions for the new node. It evaluates the resulting overall semantic distance after the insertion and selects the one that minimizes the overall semantic distance. A new node c_z is tried as either a parent or a child node to all existing nodes in the current browsing hierarchy. It can be added at the root, leaf, or internal

Algorithm 1: Building Browsing Hierarchy by Minimum Evolution

1. $W = \min_{W \succeq 0} \sum_{x=1}^{|N(c_{tr_x})|} \sum_{y=1}^{|N(c_{tr_y})|} (d_{c_{tr_x}, c_{tr_y}} - \sqrt{\Phi(c_{tr_x}, c_{tr_y})^T W^{-1} \Phi(c_{tr_x}, c_{tr_y})})^2$
2. **foreach** $c_z \in C \setminus S$
3. $\quad S \leftarrow S \cup \{c_z\}$
4. $\quad d(c_z, .) = \sqrt{\Phi(c_z, .)^T W^{-1} \Phi(c_z,)}$;
5. $\quad u = |\sum_{c_x, c_y \in S^{n-1} \cup \{c_z\}} d(c_x, c_y) - \sum_{c_x, c_y \in S^{n-1}} d(c_x, c_y)|$
6. $\quad v = \sum_{c_x, c_y \in P_{c_z}} d(c_x, c_y)$
7. $\quad R \leftarrow R \cup \{\arg\min_{R_{(c_z, .)}} \lambda u + (1 - \lambda) v\}$;
8. **Output** $T(S, R)$

Fig. 2. Building a browsing hierarchy by Minimum Evolution. C, S, and R denote the full node set, modified node set, and current relations, respectively. $N(c_{tr_x})$ is the neighborhood of a training node c_{tr_x}, including its parent, child, and sibling. $R(c_z, .)$ indicates the set of relations between a new node c_z and all other existing nodes. u represents the overall semantic distance (Eq. 11) and v the path semantic distance (Eq. 12). P_{c_z} is the path that contains the new concept c_z.

part of the hierarchy. When a node in the new node set is added into the browsing hierarchy, there is an increase in the overall semantic distance of the entire browsing hierarchy because it introduces several distances, which are non-negative and did not exist before, from itself to other nodes. We observe that when a node is in its correct position, it should make the *least addition* to the overall semantic distance in the browsing hierarchy when compared to all other possible placements. Thus minimizing the increase in overall semantic distance is equivalent to searching for the best possible position for a node.

In Algorithm 1, Line 1 indicates learning a PSD weight matrix W from the training data for the semantic distance function in Eq. 1. Line 2 to Line 8 present the minimum evolution framework to organize the *new nodes* U. Line 4 predicts distances for c_z and Line 7 finds the best structure after inserting c_z by minimizing two optimization objectives defined in Line 5 and Line 6. The hierarchy is gradually grown into a full browsing hierarchy.

6. MINIMIZING PATH INCONSISTENCY

At this point, a browsing hierarchy is formed. However, path inconsistency is a problem, which is mainly caused by polysemies. For example, under the node "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 nodes separately, they both seem correct. However, without special constraints, the two groups are connected because they share the same node – "sport." The resulting node chain is obviously wrong due to the inconsistency in meanings along the path.

To enforce path consistency along a path from the root to a leaf in a browsing hierarchy, we propose to require all nodes on the path to be about the same topic. They need to be coherent no matter how far away two nodes are apart on this path. The minimum evolution framework that we described is able to incorporate new constraints with ease.

We propose to minimize the sum of semantic distances in a path. In particular, when a new node c_z is added into a browsing hierarchy T, we require that the optimal root-to-leaf path $\hat{P}$ containing c_z should satisfy the following condition:

$$\hat{P_{c_z}} = \arg\min_{P'_{c_z}} \sum_{c_x, c_y \in P'_{c_z}} d(c_x, c_y), \tag{12}$$

where P_{c_z} is a root-to-leaf path including c_z.

Through imposing this constraint on the paths, we can prohibit node intransitivity in the browsing hierarchies. For example, when the distance between *financial institute* and *river bank* is big, the path *financial institute→bank→river bank* will be pruned, and the new node will be put in other places.

To handle both the overall semantic distance minimization and path semantic distance minimization simultaneously, we introduce a variable $\lambda \in [0, 1]$ to control the contributions from both objective functions. We formulate the multi-criterion optimization as:

$$\min \lambda u + (1 - \lambda)v \tag{13}$$

subject to $0 \leq \lambda \leq 1$, where u denotes the overall semantic distance (Eq. 11), and v denotes the path semantic distance (Eq. 12). We empirically set λ as 0.8. It appears as Line 6 in Algorithm 1.

Suppose the size of the node set C is N. C is sorted by the document frequency of the terms in the collection in decreasing order. Among the N nodes, m of them have already been added in the browsing hierarchy, $N - m$ nodes are yet to be added. To position a new node c_z, we place it at $2m$ temporary positions as either a dummy parent or a dummy child to an existing node in the browsing hierarchy. For each temporary position, the time complexity is $O(m)$ to compute the distances between c_z to all existing m nodes. Note that the pair-wise distances between existing nodes in the hierarchy have already been calculated in the previous iterations. Therefore they don't add cost to the algorithm. Moreover, the path consistency objective does not introduce much overhead cost for the same reason. Finally, since we grow the browsing hierarchy from scratch, m increases from 1 to $N - 1$. The overall time complexity is $O(2 * 1^2 + 2 * 2^2 + ... + 2 * (N - 1)^2) = O(\frac{1}{3}(N - 1)N(2N - 1)) = O(N^3)$. Thus the big O notation for Algorithm 1 is $O(N^3)$.

7. INTERACTIVE BROWSING HIERARCHY CONSTRUCTION

To incorporate task-specifications into the hierarchies that we build using the automatic approach, we allow users to directly edit the hierarchy. Here we assume that our user is someone who would like to organize the documents quickly through a combination of automated methods and some light-weight human interventions.

The interactive browsing hierarchy construction process is shown below:

(1) Extract nodes from the document collection (Section 3).
(2) Automatically create an initial browsing hierarchy as in Section 4 and Section 5. Present it to the user.
(3) The user modifies the initial hierarchy with a few edits (say, two to five); the system collects the modifications from the user (Section 7.1).
(4) Based on the user's modifications, learn a distance metric function. Use the new distance function to predict semantic distances among nodes that have not been modified by the user. Update the browsing hierarchy according to the new semantic distances and the optimization framework. Present the updated browsing hierarchy to the user. (Section 7.2)
(5) Repeat steps 3 and 4 until the user is satisfied with the browsing hierarchy.

Figure 3 shows the user interface of our browsing hierarchy construction tool. The tool's name is 'OntoCop,' which stands for "Ontology Construction Panel." The interface supports editing functions such as dragging and dropping, adding, deleting, and renaming nodes. When a user puts node c_x under c_y, it indicates that the user wants a relation demonstrated by $c_x \rightarrow c_y$ to be true in this browsing hierarchy. The user can achieve this by either dragging or dropping c_y, or by adding it as a new node under c_x.

We capture user inputs as *manual guidance*. It is used as training data to adjust our supervised distance learning, and to predict new distance scores for nodes the user has not touched yet. The user has no need to complete all the modifications he/she wants in one interaction cycle; instead, she can enter a few modifications, say two to five drag-and-drop actions, and click the *interact* button to trigger the learning algorithm. Every time the *interact* button is pressed – which is like taking a snapshot – the current partial browsing hierarchy is sent to the algorithm for analysis, to create training data.

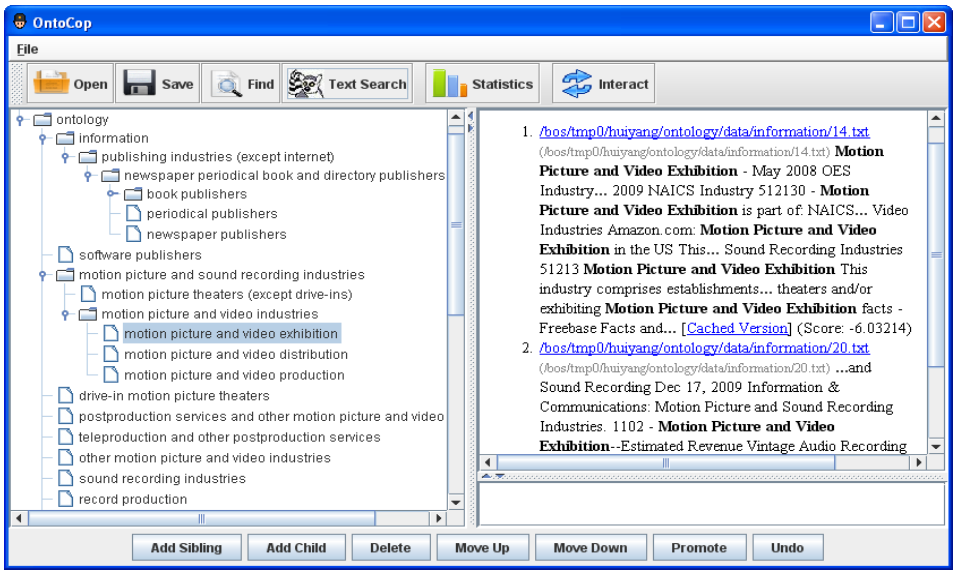


Fig. 3. OntoCop, A tool for browsing hierarchy construction.

7.1. Learning from Manual Guidance

A challenge in this interactive process is to represent human modifications in a format that can be easily understood by the distance-learning algorithm. We propose to "take snapshots" of the browsing hierarchy before and after each human-computer interaction cycle, and convert the hierarchy snapshots into matrices. The matrices indicate confidence levels of the present relations in the hierarchy snapshots. The differences in the matrices between the snapshots are used to derive *manual guidance*.

In particular, we represent the organization of nodes before a set of user modifications in between two snapshots as a *before matrix*; likewise, the new organization of nodes after the modifications is represented as an *after matrix*. The $(x, y)^{th}$ entry of a *before matrix* or an *after matrix* indicates how confident we are that a relation $r(c_x, c_y)$ is true. It is a real number between 0 and 1. r could be any type of relation between the nodes. For example, if the matrix represents an asymmetric relation in the browsing hierarchy, such as *is-a* and *part-of*, an edge weight of 0.7 is assigned to edges indicating sibling relations, an edge weight of 1 is assigned to edges indicating $parent \rightarrow child$, while an edge weight of -0.5 is assigned to edges indicating $child \rightarrow parent$. If the matrix represents a symmetric relation, such as *sibling* and *antonym*, an edge weight of 1 is assigned to edges indicating symmetric relations while 0 is assigned to other edges. Figure 4 shows an example browsing hierarchy's before and after matrices for *sibling* relations with unchanged node sets.

We define *manual guidance* M as a submatrix that 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. That is,

$$M = B[row; col],$$

where $row = \{i : b_{ij} - a_{ij} \neq 0\}$, $col = \{j : b_{ij} - a_{ij} \neq 0\}$, a_{ij} is the $(i, j)^{th}$ entry in A, and b_{ij} is the $(i, j)^{th}$ entry in B. Note that manual guidance is not simply the matrix difference between A and B. It is part of the *after matrix* that indicates where the user wants the browsing hierarchy to develop. The manual guidance for the example shown in Figure 4 is: $M = B[2, 3, 4; 2, 3, 4] =$

	oscars	*best supp.*	*best pict.*
ocsars	1	0	0
best supp.	0	1	1
best pict.	0	1	1

.

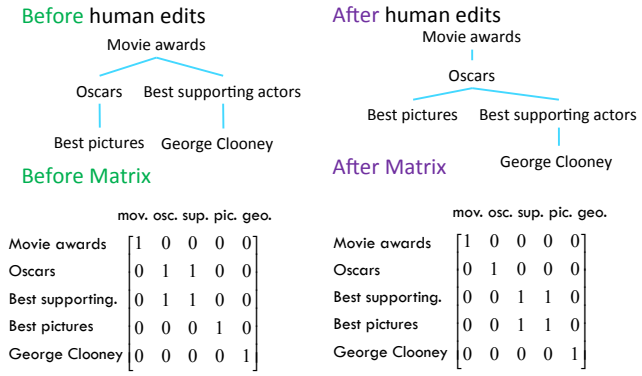


Fig. 4. An example browsing hierarchy before and after human edits (No node addition or deletion performed by the user; relation type = $sibling$); moved the node "best supporting actors" from "movie awards" to "Oscars." 1 indicates two nodes are siblings to each other. 0 indicates they are not. Note that for simplicity, we show only 1 and 0 to indicate the distance. However, the numbers can be real numbers between 0 and 1.

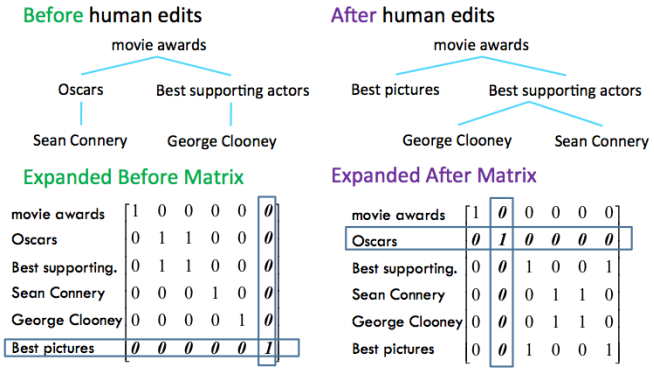


Fig. 5. An example hierarchy before and after human modifications (Node set changes; relation type = $sibling$).

When the node set changes – i.e., when the user adds or deletes nodes – we can expand rows and columns in A and B by filling 0 for non-diagonal entries and 1 for diagonal entries (as in Figure 5). Suppose the node set in the hierarchy before the modifications is C_A, and the node set after modifications is C_B. We define an expanded set of nodes C_E as the union of C_A and C_B: $C_E = C_A \bigcup C_B$.

For hierarchies with node set changes, the manual guidance M is defined as a submatrix that 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 nodes corresponding to these entries should exist in C_B, the unexpanded set of nodes after human modifications. Therefore,

$$M = B[row; col]$$

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

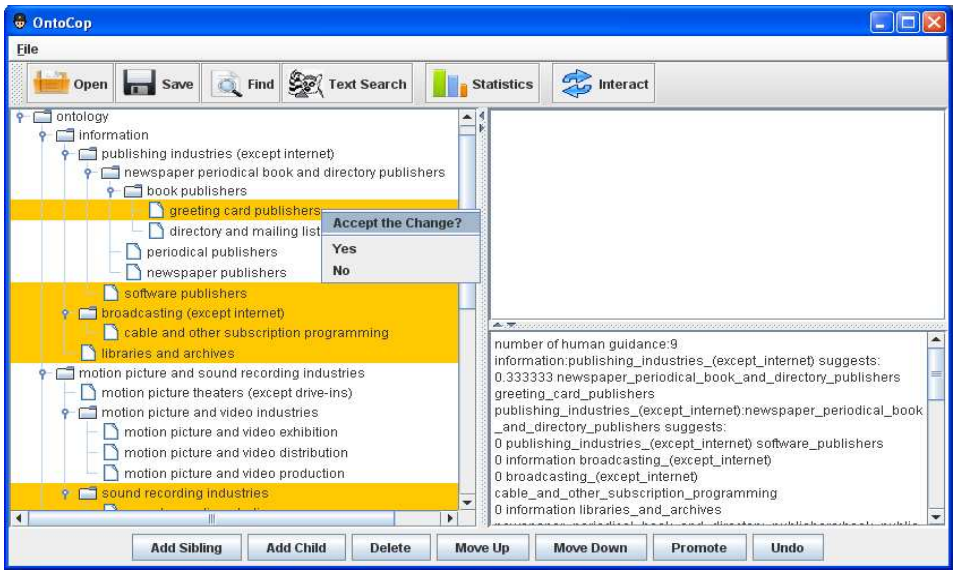


Fig. 6. An updated browsing hierarchy with new system suggestions highlighted.

Figure 5 shows an example browsing hierarchy before and after matrices for *sibling* relations with changed node sets. Here the manual guidance M is: $M = B[2,3,4,5;2,3,4,5] = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} =$

	Bestsupp.	*Sean*	*George*	*Bestpictures*
Bestsupp.	1	0	0	1
Sean	0	1	1	0
George	0	1	1	0
Bestpictures	1	0	0	1

.

7.2. Updating the Browsing Hierarchy

After learning a new distance metric function from manual guidance, we can apply it to predict pairwise distances for the *unmodified nodes* U. The *unmodified nodes* are nodes that the user has not explicitly touched (dragged or dropped). However, they should be organized in a way that follows the user's organizing philosophy – i.e., task specification or personal preferences – which we hope can be revealed by the user edits. We estimate the unmodified nodes' semantic distances using the distance functions learned from the user, and organize them into proper positions in the hierarchy.

In practice, the user only provides a few manual guidance actions in each human-computer interaction cycle; hence the training data might be too sparse for an iteration. This may cause overfitting in supervised distance learning. To avoid overfitting and too-rapid change of the hierarchical structure, we employ background training collections – WordNet [Fellbaum 1998] and ODP [ODP 2011] – to smooth and formulate better models with less variance. We collected 100 hypernym fragments from WordNet and ODP as the training data. We use the same weighting scheme as for the *before* and *after matrices* in Section 7.1, and generate more training data by summing edge weights along the shortest path between nodes. Let D be the set of node pairs that are indicated by manual guidance. The ratio of training data between D and existing hierarchies is empirically set to 1:5. We randomly select node pairs from WordNet and ODP fragments to meet this ratio.

Overall, we formulate the distance learning model with smoothing as:

$$\min_{W} \sum_{x=1}^{|\mathbf{D+WN+ODP}|} \sum_{y=1}^{|\mathbf{D+WN+ODP}|} \left(d_{c_x,c_y} - \sqrt{\Phi(c_x,c_y)^T W^{-1} \Phi(c_x,c_y)} \right)^2 \tag{14}$$

subject to $W \succeq 0$, where D is the set of nodes edited manually, ODP is the set of nodes from ODP, and WN is the set of nodes from WordNet.

The unmodified nodes are now inserted into their optimal positions based on the minimum evolution framework. The construction tool presents the modified browsing hierarchy to the user, with highlights in the part changed by the system. Figure 6 shows an updated browsing hierarchy with system changes highlighted. The user can judge the correctness of these machine-generated modifications. They show up as new nodes or repositioned nodes in the hierarchy, and look no different from other nodes. Judging a suggestion can be done by selecting an option "yes" or "no" from the "Accept the change?" menu.

The system then waits for the next round of manual guidance. This human-teaching-machine-learning process continues until the user is satisfied with the hierarchy.

The proposed human-guided learning method accepts direct user edits in the form of grouping the nodes, splitting the nodes, moving the nodes, and introducing new nodes in the hierarchy, which are all user-centric interactions that allow the user to feel no machine intervention between them and OntoCop. To achieve this goal, we make sure that at each interaction, only a few edits are collected from the user, to avoid boredom and fatigue–in our case, three edits per interaction on average. These few manual edits are used as first-hand feedback (both positive and negative) to the learning algorithm, which adapts to them rapidly, without any smoothing. It means we will have rapidly changing hierarchy structures, which is undesirable. To attain satisfactory performance, we use training instances from existing taxonomies, WordNet and ODP, to smooth out the potential rapid change from only using manual guidance.

We did not observe an obvious learning curve showing up in the interactive experiments. User requirements for hierarchy quality have been high since the beginning. The users have expected high-quality hierarchies from the beginning, just as they expect for Web search, where they expect to see highly relevant documents ranked high on the first page. For the same reason, we use the smoothed version of human-guided learning from the beginning of the interactive construction process, instead of transitioning into using only manual guidance by population-level preferences. An interesting future exploration will be to collect manual guidance across multiple users for the same tasks, and study how population-level preferences can affect the learning process.

8. EVALUATION

We conducted experiments and a user study to evaluate the effectiveness of our approach. We had two goals for the evaluation. One is to evaluate how the browsing hierarchies constructed by our automatic approach compare with those constructed by state-of-the-art baseline systems (Sections 8.1 to 8.5). Another is to investigate how well our system can learn from task specifications from users in the interactive approach. (Section 8.6).

8.1. Datasets

The datasets we used in the evaluation include public comments and Web documents. Six public comment datasets are used for the experiments: '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). Table II 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.

The Web datasets are collections of Web documents generated during complex search tasks. We first select 50 search topics from a commercial search log and treat each topic as a search task. Example topics are *find a good kindergarten*, *purchase a used car*, *plan a trip to DC*, *how to make a cake*, *find a good wedding videographer*, *write a survey paper for health care systems*, *find the best deals for a Mother's day gift*, *write a survey paper for social network*, *write a survey paper for EU's finance*, and *write a survey paper for information technology*.

Table II. Statistics of public comment datasets.

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

We then created one dataset for each search topic, by submitting related queries to and collecting the returned Web documents from two search engines *bing.com* and *google.com*. The queries are manually selected from the search log that we used. For example, queries "trip to DC," "Washington DC," "DC," and "Washington" were submitted for the search topic "planning a trip to DC." We filter out spam and advertisements, and then search for more relevant Web documents. Around 1000 Web documents are collected for each dataset. Among all 50 datasets, the average number of documents is 988.5. The average number of unique words in a dataset is 698,875. The top 200 ranked nodes after node extraction as presented in Section 3 are kept for each dataset in experiments in Sections 8.2 through 8.4. The top 40 ranked nodes are kept for experiments in Section 8.6.

8.2. Systems Under Comparison

We compare our automated system with a clustering approach, a monothetic approach, and a linguistic approach in this evaluation. The facet-construction approach does not adapt well to the domain-specific tasks that we study here; therefore we did not compare our approach with it. The following systems are evaluated in the experiments:

— Hierarchical clustering: We employ WEKA[11] to form hierarchical document clusters, and then assign labels to the clusters. We used bottom-up hierarchical clustering based on a Euclidean distance function. The labeling is done by a state-of-the-art cluster-labeling algorithm [Carmel et al. 2009].
— Subsumption: The automatic algorithm proposed by [Sanderson and Croft 1999]. It makes use of terms' conditional probability to determine the pair-wise relationships among them. It is the most widely used browsing hierarchy construction technique.
— KH: The automatic hierarchy construction algorithm proposed by [Kozareva and Hovy 2010]. It is a state-of-the-art linguistic pattern-based approach that uses a double-anchored pattern for recognizing is-a relations and organizing the nodes into hierarchies.
— ME: The proposed automatic minimum evolution optimization framework.
— MEPath: The proposed automatic hierarchy construction algorithm with path consistency control.

8.3. Browsing Effectiveness

A popular measure to evaluate the quality of the browsing hierarchies is the expected mutual information measure (EMIM [Lawrie et al. 2001]). It calculates the mutual information between the language model in a hierarchy T and the language model in a document collection Z:

$$I(C;V) = \sum_{c \in C, v \in V} Pr(c,v) log \frac{Pr(c,v)}{Pr(c)Pr(v)}, \tag{15}$$

where $Pr(c,v) = \sum_{d \in Z} Pr(d)Pr(c|d)Pr(v|d)$, C is the set of nodes in T, V is the set of non-stopwords in Z, and d is a document in Z. $Pr(.)$ stands for probabilistic functions. EMIM only evaluates the content of a browsing hierarchy, not its structure. However, it is used to indicate how representative a browsing hierarchy is for a document collection.

Table III shows the EMIM of the browsing hierarchies constructed by the systems under evaluation. Based on the mean EMIM over all datasets, we can rank the systems in terms of EMIM

[11] http://www.cs.waikato.ac.nz/ml/weka/, version 3.6.6.

Table III. Expected Mutual Information (in 1000*EMIM)

Example datasets	Clustering	Subs.	KH	ME	MEPath
kindergarten	0.3	0.4	3.8	3.9	5.6
health care	0.3	0.5	2.8	3.1	7.8
used car	0.2	0.1	0.2	0.1	2.8
trip to DC	0.2	0.2	4.3	4.5	6.4
finance	0.01	0.01	0.01	0.1	0.6
gift	0.1	0.2	1.2	1.2	3.8
social network	0.05	0.1	1.5	1.3	2.4
information	0.2	0.3	1.9	2.3	3.5
cake	0.2	0.2	1.2	3.1	6.6
videographer	0.4	0.4	1.8	1.6	4.9
Mean of all datasets	**0.22**	**0.24**	**1.7**	**2.1**	**4.4**

Table IV. Reach time

Example dataset	Clustering	Subs.	KH	ME	MEPath
kindergarten	15.7	14.4	9.8	9.9	8.7
health care	13.8	12.3	9.8	6.8	4.5
used car	16.1	15.4	12.4	10.2	8.7
trip to DC	13.5	11.2	10.3	9,8	8.7
finance	26	24.5	18.3	19.7	18.7
gift	12.7	11.2	8.4	7.7	5.6
social network	15.7	14.3	9.8	7.8	7.6
information	11.1	10.6	9.5	8.8	8.9
cake	9.2	8.9	4.8	4.5	3.4
videographer	9.3	9.5	8.8	7.6	6.9
Mean of all datasets	**15.6**	**14.2**	**12.2**	**9.8**	**7.2**

in descending order as MEPath $>>$ ME $>$ KH $>>$ Subsumption $>$ Clustering. $>>$ indicates a statistically significant difference ($p < .001$, one-sided t-test) and $>$ indicates moderate statistical significance ($p < .05$, one-sided t-test). We will use the same symbols throughout the remainder of this paper. It shows that MEPath is the best performing $automatic$ algorithm to generate browsing hierarchies. MEPath is 109% as effective as ME (p-value$<$.001, one-sided t-test), 159% as effective as KH (p-value$<$.001, one-sided t-test), and 17 times as effective as subsumption (p-value$<$.001, one-sided t-test). All of these represent statistically significan differences. This strongly suggests that our techniques are more effective than the state-of-the-art systems in constructing browsing hierarchies.

Another popular evaluation measure[12] for browsing effectiveness is *reach time* [Carpineto et al. 2009]. It is defined as:

$$t_{reach} = \frac{1}{|Rel|} \sum_{d_i \in Rel} L(c_i) + pos_i \tag{16}$$

where Rel is the relevant documents, c_i is the node that connects to a relevant document d_i, $L(c_i)$ is the path length from the root to reach c_i, and pos_i is the position where d_i appears in the document cluster associated with c_i. The minimum path to a document is used if multiple paths exist. Reach time evaluates both the content and the structure of a browsing hierarchy. This measure needs relevance judgments about a query for the documents organized by the hierarchies. We obtained relevance judgments by using the majority votes from a user study involving 29 subjects, followed by expert reviews. Three experts manually examined the majority votes and reached agreement on all relevance judgments.

Table IV reflects reach time for each of the systems. Based on the mean reach time over all datasets, we obtain a similar ranking of the systems to that suggested by EMIM. The ranking based on reach time is: MEPath $>>$ ME $>$ KH $>$ Subsumption $>$ Clustering. It shows that the best-performing $automatic$ system is MEPath, which on average can produce hierarchies to reach a

[12]Other proposed measures include coverage and compactness [Kummamuru et al. 2004].

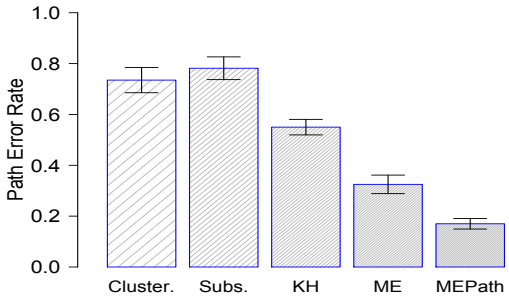


Fig. 7. Path error rate.

relevant document by visiting only 7.2 nodes, including 5.2 non-leaf nodes and 2 documents in the leaf cluster, on average. To find all relevant documents in a collection sized around 1000, this reach time is very fast.

Both EMIM and reach time demonstrate that the proposed minimum evolution approach is able to build very effective browsing hierarchies for a document collection.

8.4. Path Consistency

To evaluate how well path consistency is handled, we compare the path error rate generated by our approach and by other baseline systems.

The path error is defined as the average number of wrong ancestor-descendant pairs in a hierarchy. It is only applied to nodes *not* immediately connected. It can be judged and calculated as follows. Three human assessors manually evaluated the path errors by (1) gathering the paths by performing a depth-first traverse in the hierarchy from the root node; (2) along each path, counting the number of wrong ancestor-descendant pairs; (3) summing up the errors that all assessors agree on and normalizing the sum by the hierarchy size.

Figure 7 shows the path error rate generated by all the automated algorithms under evaluation. We can see that MEPath produces the least path error. The algorithms can be ranked in terms of the ability to handle path consistency as MEPath $>>$ ME $>>$ KH $>$ Clustering $>>$ Subsumption. MEPath reduces path errors over not using the path consistency control (ME) by 500% (p-value$<$.001, one-sided t-test), a statistically significant outcome. It strongly indicates that our technique is effective to maintain path consistency.

8.5. Features vs. Relations

To study the impact of different types of features on different types of relations, we group more than 30 features used in our framework into six sets: contextual, co-concurrence, patterns, syntactic dependency, and word-length difference and definition, and turn on each feature set one by one.

Table V shows the accuracy of reconstructing existing hierarchies using each set of features alone in automatic hierarchy construction for WordNet *is-a, sibling,* and *part-of* relations. We use subtrees from WordNet and ODP as the ground truth[13]. Bold font indicates that a feature set makes a statistically significant contribution ($p < 0.005$, one-sided t-test) to automatic browsing hierarchy construction for a particular type of relation.

[13]WordNet *is-a* hierarchies are from 12 topics: gathering, professional, people, building, place, milk, meal, water, beverage, alcohol, dish, and herb. ODP *is-a* hierarchies are from 16 topics: computers, robotics, intranet, mobile computing, database, operating system, linux, tex, software, computer science, data communication, algorithms, data formats, security, multimedia, and artificial intelligence. WordNet *part-of* hierarchies are from 15 topics: bed, car, building, lamp, earth, television, body, drama, theater, water, airplane, piano, book, computer, and watch.

Table V. 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		

The metrics used for hierarchy comparison are the F1-measure for node pairs and fragment-based similarity (FBS) [Yang 2011; 2014]. Given two hierarchies H_i and H_j, FBS compares their similarity by (1) first converting each internal node c into a vector of 1's and 0's to indicate presence or absence of other nodes in the subtree led by c, (2) matching the internal nodes in T_i with those in T_j by cosine similarity between the nodes, and (3) aggregating the overall similarity scores for the matched nodes. The metric FBS is

$$\frac{1}{max(I,J)} \sum_{p=1}^{m} \text{Sim}_{cos}(c_{ip}, c_{jp}), \tag{17}$$

where $c_{ip} \subseteq T_i$, $c_{jp} \subseteq T_j$, which are the p^{th} matched pair among the m matched fragment pairs and I (J) is the number of nodes in T_i (T_j).

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 well 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 four out of five syntactic features (*Modifier*, *Subject*, *Object*, and *Verb* overlaps) are actually surrounding context for a node.

The second to last row of Table V show 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 ($p < 0.001$, one-sided t-test). It indicates that combining heterogeneous features gives significant rise to the system performance than any single type of feature does.

8.6. User Study

Besides objective evaluations, we conducted a within-subject user study to evaluate the interactive version of our system. OntoCop, the hierarchy construction tool presented in Section 7 (Figure 3), is the main tool evaluated in this user study. We compare the interactive setting of OntoCop to a non-interactive setting, where the users use OntoCop to refine the automated constructed hierarchy without the system learning from manual edits. We also use questionnaires to get user ratings of our system.

Twenty-nine (thirty initially; one was excluded because of incomplete data entry) graduate and undergraduate students from various majors in three universities participated in the study. The ages of the participants ranged from 19 to 33 years old (Mean=24.3, SD=2.38). Thirteen participants were female (45%) and sixteen were male (55%). 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 speakers (66%) or had high proficiency (34%) in English. Table VI summarizes the statistics of the participants in the user study.

Table VI. Statistics of participants

Gender	
Male	16
Female	13
Majors	
Arts	3
Business	1
Chemical Engineering	2
Cognitive Science	2
Computer Science	8
Math	1
Public Policy	8
Psychology	4
English Proficiency	
Native speaker	19
Non-native speaker (with high proficiency)	10

The participants were introduced to the hierarchy construction user interface, and given about 10 minutes to get familiar with its functions. This training was then followed by an exercise task that lasted about 15 minutes. Afterwards, the participants started the real tasks. Once the real tasks were done, participants had 10 minutes to answer a questionnaire regarding their experience.

For each task, participants used OntoCop to construct a browsing hierarchy for a list of given nodes. We asked the participants to construct browsing hierarchies with the following task in mind.

> Imagine you have a task [task name]. You are building a browsing hierarchy designed for the collection of Web documents about this task. Later on, you will use the browsing hierarchy to find all useful topics for your task. You will need to identify at least one document for each topic.

We asked the participants to "find all useful topics" because we consider that as a browsing hierarchy, it should support broad topic coverage in the document collection.

The size of each node set in the user study was set to 40, to allow the participants to be able to finish a task in a reasonable amount of time. The participants had access to the documents that are linked to the nodes so that they could read them while constructing the hierarchy. Each participant had a maximum of 25 minutes for each real task. The participants were not required to finish all tasks. However, a minimum of four datasets was required of every participant. The datasets and algorithms under evaluation in the user study were distributed in a round-robin fashion, to minimize the learning effects on our results.

Each participant had a chance to test two settings: non-interactive and interactive. In a non-interactive setting, the participant used OntoCop to organize the hierarchy constructed from what was constructed by the automated approach (Section 5), without using the 'interact' function. In the interactive setting, the participant used the 'interact' button to send manual guidance to the system to refine the hierarchies. The participant could add new nodes, drag & drop nodes, or delete nodes, renaming nodes at any point.

The following sections report the findings from the user study.

8.6.1. Accuracy of System Predictions. To evaluate how well the proposed interactive approach can learn from user-generated task specifications, we measured the accuracy of system predictions by engaging the participants to judge how well the system learns from human edits through a direct evaluation during the process.

Table VII shows the statistics of the participants' editing activity, including the statistics for our interactive learning run, and the statistics for a non-interactive run that builds hierarchies without the algorithm learning from the user. The types of editing that a participant can perform when constructing browsing 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 VII. 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*
Non-interactive 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*

Table VIII. Accuracy of system predictions.

	Max	Min	Avg
accuracy of system predictions	0.98	0.92	0.94

Table IX. Perceived learning ability

	Max	Min	Avg
perceived learning ability	4.2	2.8	3.61
which dataset	health care	finance	-

The table shows that "drag & drop" is the most popular editing function. More than 90% of time, the participants used the "drag & drop" function to move the nodes around to construct the browsing hierarchies. It is probably due to the ease of using this function and their familiarity with this function through their past computer-use experience.

When a user provided manual guidance to the interactive system, during each human-computer interaction cycle, the system made predictions based on the user's edits. He or she could directly judge the correctness of these machine-predicted modifications on-the-fly by selecting an option "yes" or "no" from the "Accept the change?" menu. Note that these were personalized tasks and the predictions were evaluated by the user according to his/her own standard. We calculate the accuracy of system predictions as:

$$\text{Accuracy} = \frac{1}{\text{total \# of interaction cycles}} \sum_i \frac{\text{\# of accepted predictions in the } i^{th} \text{ cycle}}{\text{\# of predictions in the } i^{th} \text{ cycle}},$$

where I is the total number of human-computer interaction cycles when constructing a browsing hierarchy. A high accuracy indicates that the system learns well from the user edits.

The modifications made by the system (on which accuracy was assessed) show up in the hierarchy with highlights. They are usually new added terms in a group, or a new group that is formed and added into the hierarchy.

Table VIII shows that for all datasets, the mean accuracy of the system predictions is above 92%. The average value is 94%. This high accuracy clearly demonstrates that the system successfully learns from a user and makes highly accurate predictions on how the user would organize the nodes. Moreover, the mean number of system predictions across all datasets is 15.2. It indicates that about 38% of the relations in a finalized browsing hierarchy were suggested by the system, and among them at least 92% were accepted by the participants.

8.6.2. Perceived Learning Ability. After constructing a browsing hierarchy, each participant was immediately asked the following question *"How well did the browsing hierarchy help you to complete the task?"*. Given the nature of hierarchy construction, it is more difficult to build a good hierarchy than retrieving a relevant document *ad hoc*. In other words, it is much harder to obtain a "perfect" hierarchy than a "perfect" retrieved document within a poor hierarchy. We therefore designed a five-point rating scale that is more refined towards the positive side of the evaluation. Our ratings are "very good"(5), "good"(4), "fair"(3), "bad"(2), and "trash"(1); they are used to rate the perceived system learning ability.

Table IX shows the max, min, and average responses of perceived system learning ability. The mean perceived learning ability is 3.61, with a standard derivation of 0.45. This suggests that the learning ability of the system was only perceived as moderately good. This result contradicts the

conclusion that we drew based on the more objective measure, accuracy of system prediction (Section 8.6.1). To find out why, we investigated why the participants were only moderately satisfied with the system's learning ability. From the after-session questionnaire, we found that participants thought that some datasets such as "finance" were more difficult than other datasets, such as "health care." For example, the dataset "finance" was considered by all participants "very difficult," while "health care" was considered "very easy." The participants also complained that they were not familiar with the difficult datasets. It is interesting that when a dataset is less familiar for the users, the system was perceived to be performing badly too. It may suggest that when people are not familiar with the tasks, they provide less promising edits; then the system learns from the lower-quality training data, and in the end the users perceive the output as poor system learning ability.

9. DISCUSSION

In this paper, we propose a novel hierarchy construction approach, which supports flexibly adding or removing constraints, including user-defined ones, that describe the relationships among the nodes in a hierarchy and precisely define how the hierarchy should be constructed. We uniquely employ pair-wise semantic distance as an entry point to incrementally build browsing hierarchies. The supervised distance learning algorithm not only allows us to incorporate multiple semantic features to describe the proximity between nodes, but also allows us to learn a metric function from task specifications. It allows our approach to build task-specific browsing hierarchies for more complex search tasks. Users can thus manually modify the hierarchies and to some extent teach the algorithm to predict his/her way of organizing the nodes.

In addition, by minimizing the overall semantic distances among nodes and restricting minimal semantic distances along a path, we find the best hierarchical structure for the browsing hierarchy. It tries to put together semantically close nodes so that users will have a good idea about why the nodes are put together. This greatly increases the *interpretability* of a constructed browsing hierarchy over most existing approaches.

In this section, we foster a few discussions on the implementation details, reasons why we made such choices, and some unsolved issues that will be explored in future work.

9.1. Order of adding nodes

It is worth noting that given all pair-wise distances, finding a tree with the smallest edge weight sum is equivalent to the minimum spanning tree problem (MST). However, we cannot directly apply existing solutions for tree construction algorithms to our problem. This is because browsing hierarchy construction is a task that involves natural language. Building a hierarchy for natural language documents needs to satisfy many constraints, due to the language's richness and complex structures. Finding an optimal tree structure satisfying certain constraints is often intractable. In this work, we show how to incorporate constraints into a tree-structure construction iteratively.

The order of adding the nodes may affect the final hierarchical structure. The basic order of adding the nodes is based on their sorted document frequency order. That is to say, nodes that have higher document frequency are added first into the hierarchy.

We introduce some randomness in the order of adding nodes. We add the nodes with γ random restarts and select the hierarchy that minimizes the objectives among all candidates. The value of an effective γ is related to the size of the hierarchy size. In our experiments, the node set is kept in the range of 40 to 200. We therefore were able to set γ=10 in our experiments. It does not entirely solve the local minimum problem; however, it allows us to reach a balance between accuracy and speed.

We are interested in investigating the stability of the hierarchies with different random restarts. To quantitatively evaluate the stability of hierarchies, we compare the similarity among hierarchies created for one task with different random restarts. The stability of γ random restarts is measured by the average of pairwise hierarchy similarity between unique query pairs. It is defined as: $Stability(\mathcal{H}) = \frac{2}{\gamma(\gamma-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \text{Sim}_{hie}(H_i, H_j)$, where γ is the number of random restarts, H_i and H_j are two hierarchies, and $\text{Sim}_{hie}(H_i, H_j)$ is the hierarchy similarity between H_i and

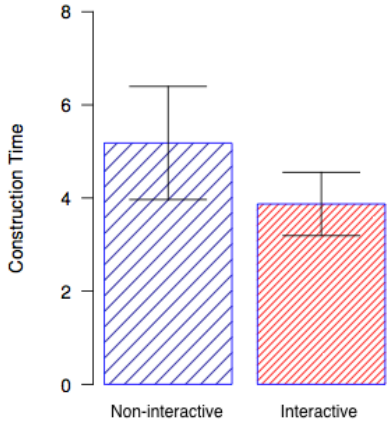


Fig. 8. Mean hierarchy construction time (in minutes; both non-interactive and interactive runs)

H_j [Guan and Yang 2013]. Sim_{hie} can be calculated by fragment-based similarity (FBS) (Eq. 17) [Yang 2014]. FBS considers both content and structure differences; it measures differences at fragment level and tolerates minor changes in hierarchies. The experiments show that the stability of hierarchies with different random restart ranges from 0.7 to 0.9, which is high. It suggests that the randomness introduced by the random restarts does not have a negative effect on the effectiveness of the proposed automated method for hierarchy construction.

9.2. Multiple inheritance

We do not support multiple inheritance in our system. The current framework only chooses a single "best" position for each node. The consequence for homonyms–words spelled the same that have different meanings–is that one of them will be removed from the hierarchy. For example, after identifying the inconsistency between "financial institute $\rightarrow$ bank $\rightarrow$ river bank," one of the meanings for "bank" is lost.

We argue that the best form of organizing information is multiple tree representations, with additional links between nodes in a tree. We address this issue by having the users participate in an interactive process and taking into account their preferences, so that each browsing hierarchy is one tree with one specific view to the data. In this paper, we do not study how to allow additional links in a browsing hierarchy.

In order to introduce additional links into a tree, we must support multiple inheritance. Some nodes could have multiple parents, such as "bank": it is a "financial institute" and it is also a part of a "river." For instance, we can train classification models to predict if a node should be positioned into multiple locations in the browsing hierarchy. Given a gold-standard browsing hierarchy, we can extract features to determine if a node 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 resources and training data to achieve reasonable performance.

9.3. Scalability

The majority of our work has been carried out on organizing the nodes and putting them into the right positions, which we think is largely missing from most data-driven approaches. Using clustering as an example, we often find that many labeled clusters in a hierarchy do not seem 'correct.' It could be that they have the wrong labels for their clusters, that they are misplaced, that they should be further grouped or further split, etc. It is difficult to define what is 'correct,' though. Our approach emphasizes this matter and incorporates constraints to ensure the hierarchy 'looks cor-

rect.' With some degrees of loss to scalability, we achieve better 'accuracy' for browsing hierarchy construction. Our approach can be more suitable for small-scale collections such as search results during a search task, where small but 'correct' hierarchies are desirable.

In Section 5, we lay out the "big O" notation of the learning algorithm. In addition, we are also interested in wall-clock time. Figure 8 shows the average time (and its 95% confidence interval) used to construct a browsing hierarchy. For the interactive runs, the average construction time that the participants used was 3.87 minutes. For the non-interactive runs, the average construction time was 5.18 minutes. We performed statistical significance tests to analyze the construction time. For a hierarchy with 40 nodes in the evaluation, the speed of learning from user feedback in the interactive setting is adequate. Moreover, the experimental results show that the interactive method used significantly less time (one minute or 20% less per dataset on average) than the non-interactive construction method ($p < .001$ on a one-way ANOVA test, $df = 1$, $F = 11.27$).

9.4. Future Work

Accuracy comes at a cost. Our method focuses more on improving the hierarchy accuracy, less on efficiency. In this work, we limit the size of the hierarchy by only working on a portion of the nodes from the collection. In this way, we reduce the size of the hierarchy and make sure that the browsing hierarchy we build as the end product is accurate. How to scale up the proposed method for larger hierarchies is one of the future directions that we would like to explore.

Our online learning algorithm is one of the early efforts to study how to quickly learn from the user to organize nodes and documents into hierarchies. Limited by the size of the datasets used in the user study, the collected data did not have the chance to show a learning curve in the process. However, we do believe that for a complex information seeking task, the user is engaged in a learning process and some kind of learning curve is expected in their browsing hierarchy construction process, if they are using our tool. It is another future research direction that we will pursue.

10. CONCLUSION

A browsing hierarchy can supply an "information map" while information seeking activities are going on for a complex search task. The hierarchy must suit the specific domain in order to support such task-specific document organization and browsing. This paper explores techniques to derive *task-specific* hierarchies supporting browsing in a document set for complex search. The majority of our work has been devoted to organizing the nodes, with a focus on putting them into the right positions to make the hierarchy "look correct" – a process we think is missing in most data-driven approaches.

Our 'minimum evolution' browsing hierarchy construction is a novel automatic framework. It mimics how people organize items and turns browsing-hierarchy construction into an incremental process. In this process, every step of adding a new node into the browsing hierarchy is transformed into an optimization problem. For each pair of nodes 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.We believe our work is the first piece of work that seamlessly integrates features from both families of statistical measures and linguistic patterns.

Incorporating task specifications in browsing-hierarchy construction is challenging. The interactive version of the framework allows a user to provide periodic manual guidance through interaction with a learning algorithm to produce a browsing hierarchy. Through human-computer interaction, the human and the machine work together to organize nodes into browsing hierarchies. The algorithm uses the manual guidance represented by these matrices to train new browsing-hierarchy construction models, which adapt to the user's preferences. In this way, the user is "put into the loop" and is able to build browsing hierarchies. The user study and experimental results strongly suggest that the proposed techniques are highly effective in constructing browsing hierarchies.

We believe our minimum evolution framework and the online learning algorithm make a unique contribution to the field of browsing hierarchy construction. Our methods also provide novel and

practical solutions to building task-specific "information map" in the bigger context of information seeking and information triage.

ACKNOWLEDGMENTS

The author sincerely thanks Prof. Jamie Callan for in-depth discussions and editors and anonymous reviewers for their valuable comments. The research was supported by NSF grant IIS 0704210, NSF grant CNS-1223825, and the DARPA Memex program. This material is based on research sponsored by DARPA under agreement number FA8750-14-2-0226. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

REFERENCES

Ibrahim Adepoju Adeyanju, Dawei Song, M-Dyaa Albakour, Udo Kruschwitz, Anne De Roeck, and Maria Fasli. 2012. Adaptation of the Concept Hierarchy Model with Search Logs for Query Recommendation on Intranets. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '12)*. ACM, Portland, Oregon, USA, 5–14. DOI: http://dx.doi.org/10.1145/2348283.2348288

Leif Azzopardi. 2014. Modelling Interaction with Economic Models of Search. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '14)*. ACM, Gold Coast, Queensland, Australia, 3–12. DOI: http://dx.doi.org/10.1145/2600428.2609574

Jerome R. Bellegarda, John W. Butzberger, Yen-Lu Chow, Noah B. Coccaro, and Devang Naik. 1996. A Novel Word Clustering Algorithm Based on Latent Semantic Analysis. In *Proceedings of the Acoustics, Speech, and Signal Processing, 1996. On Conference Proceedings., 1996 IEEE International Conference - Volume 01 (ICASSP '96)*. IEEE Computer Society, Atlanta, Georgia, USA, 172–175. DOI: http://dx.doi.org/10.1109/ICASSP.1996.540318

Matthew Berland and Eugene Charniak. 1999. Finding Parts in Very Large Corpora. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics (ACL '99)*. Association for Computational Linguistics, College Park, Maryland, USA, 57–64. DOI: http://dx.doi.org/10.3115/1034678.1034697

Rajendra Bhatia. 2006. *Positive definite matrices (princeton series in applied mathematics)*. Princeton University Press.

David Carmel, Haggai Roitman, and Naama Zwerdling. 2009. Enhancing Cluster Labeling Using Wikipedia. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '09)*. ACM, Boston, Massachusetts, USA, 139–146. DOI: http://dx.doi.org/10.1145/1571941.1571967

Claudio Carpineto, Stefano Mizzaro, Giovanni Romano, and Matteo Snidero. 2009. Mobile Information Retrieval with Search Results Clustering: Prototypes and Evaluations. *J. Am. Soc. Inf. Sci. Technol.* 60, 5 (May 2009), 877–895. DOI: http://dx.doi.org/10.1002/asi.v60:5

Claudio Carpineto and Giovanni Romano. 2010. Optimal Meta Search Results Clustering. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '10)*. ACM, Geneva, Switzerland, 170–177. DOI: http://dx.doi.org/10.1145/1835449.1835480

Barbara S. Chaparro, Veronica D. Hinkle, and Shannon K. Riley. 2008. The Usability of Computerized Card Sorting: A Comparison of Three Applications by Researchers and End Users. *Journal of Usability Studies* 4, 1 (2008), 31–48.

Philipp Cimiano and Johanna Wenderoth. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*. 888–895.

W. Bruce Croft, Donald Metzler, and Trevor Strohman. 2004. *Search Engines: Information Retrieval in Practice*. Addison Wesley.

Carolyn J. Crouch. 1988. A Cluster-based Approach to Thesaurus Construction. In *Proceedings of the 11th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '88)*. ACM, Grenoble, France, 309–320. DOI: http://dx.doi.org/10.1145/62437.62467

Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. 1992. Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '92)*. ACM, Copenhagen, Denmark, 318–329. DOI: http://dx.doi.org/10.1145/133160.133214

Wisam Dakka, Panagiotis G. Ipeirotis, and Kenneth R. Wood. 2005. Automatic Construction of Multifaceted Browsing Interfaces. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management (CIKM '05)*. ACM, Bremen, Germany, 768–775. DOI: http://dx.doi.org/10.1145/1099554.1099738

Adriel Dean-Hall, Charles L. A. Clarke, Jaap Kamps, Paul Thomas, and Ellen Voorhees. 2012. Overview of the TREC 2012 Contextual Suggestion Track. In *Proceedings of the 21st Text REtrieval Conference (TREC '12)*. NIST, Gaithersburg, Maryland, USA.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2005. Unsupervised Named-entity Extraction from the Web: An Experimental Study. *Artif. Intell.* 165, 1 (June 2005), 91–134. DOI: http://dx.doi.org/10.1016/j.artint.2005.03.001

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Roxana Girju, Adriana Badulescu, and Dan Moldovan. 2003. Learning Semantic Constraints for the Automatic Discovery of Part-whole Relations. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1 (NAACL '03)*. Association for Computational Linguistics, Edmonton, Canada, 1–8. DOI:http://dx.doi.org/10.3115/1073445.1073456

Dongyi Guan and Hui Yang. 2013. Increasing Stability of Result Organization for Session Search. In *Proceedings of the 35th European Conference on Advances in Information Retrieval (ECIR'13)*. Springer-Verlag, Moscow, Russia, 471–482. DOI:http://dx.doi.org/10.1007/978-3-642-36973-5_40

Zelig Harris. 1970. Distributional structure. In *Papers in Structural and Transformational Linguistics*. D. Reidel Publishing Company, Dordrecht, Holland, 775–794.

Marti A. Hearst. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 2 (COLING '92)*. Association for Computational Linguistics, Nantes, France, 539–545. DOI:http://dx.doi.org/10.3115/992133.992154

Katja Hofmann, Shimon Whiteson, and Maarten de Rijke. 2011. Balancing Exploration and Exploitation in Learning to Rank Online. In *Proceedings of the 33rd European Conference on Advances in Information Retrieval (ECIR'11)*. Springer-Verlag, Dublin, Ireland, 251–263. http://dl.acm.org/citation.cfm?id=1996889.1996922

Katja Hofmann, Shimon Whiteson, and Maarten Rijke. 2013. Balancing Exploration and Exploitation in Listwise and Pairwise Online Learning to Rank for Information Retrieval. *Inf. Retr.* 16, 1 (Feb. 2013), 63–90. DOI:http://dx.doi.org/10.1007/s10791-012-9197-9

Yifen Huang and Tom Mitchell. 2007. A Framework for Mixed-Initiative Clustering. In *North East Student Colloquium on Artificial Intelligence (NESCAI 2007)*.

Evangelos Kanoulas, Ben Carterette, Mark Hall, Paul Clough, and Mark Sanderson. 2013. Overview of the TREC 2013 Session Track. In *Proceedings of the 22nd Text REtrieval Conference (TREC '13)*. NIST, Gaithersburg, Maryland, USA.

Weimao Ke, Cassidy R. Sugimoto, and Javed Mostafa. 2009. Dynamicity vs. Effectiveness: Studying Online Clustering for Scatter/Gather. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '09)*. ACM, Boston, Massachussets, USA, 19–26. DOI:http://dx.doi.org/10.1145/1571941.1571947

Diane Kelly, Amber Cushing, Maureen Dostert, Xi Niu, and Karl Gyllstrom. 2010. Effects of Popularity and Quality on the Usage of Query Suggestions During Information Search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, Atlanta, Georgia, USA, 45–54. DOI:http://dx.doi.org/10.1145/1753326.1753334

Andruid Kerne, Eunyee Koh, Vikram Sundaram, and J. Michael Mistrot. 2005. Generative Semantic Clustering in Spatial Hypertext. In *Proceedings of the 2005 ACM Symposium on Document Engineering (DocEng '05)*. ACM, Bristol, United Kingdom, 84–93. DOI:http://dx.doi.org/10.1145/1096601.1096624

Zornitsa Kozareva and Eduard Hovy. 2010. A Semi-supervised Method to Learn and Construct Taxonomies Using the Web. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP '10)*. Association for Computational Linguistics, Cambridge, Massachusetts, USA, 1110–1118. http://dl.acm.org/citation.cfm?id=1870658.1870766

Zornitsa Kozareva, Ellen Riloff, and Eduard H. Hovy. 2008. Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*. The Association for Computer Linguistics, Columbus, Ohio, USA, 1048–1056.

Krishna Kummamuru, Rohit Lotlikar, Shourya Roy, Karan Singal, and Raghu Krishnapuram. 2004. A Hierarchical Monothetic Document Clustering Algorithm for Summarization and Browsing Search Results. In *Proceedings of the 13th International Conference on World Wide Web (WWW '04)*. ACM, New York, New York, USA, 658–665. DOI:http://dx.doi.org/10.1145/988672.988762

Dmitry Lagun and Eugene Agichtein. 2011. ViewSer: Enabling Large-scale Remote User Studies of Web Search Examination and Interaction. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '11)*. ACM, Beijing, China, 365–374. DOI:http://dx.doi.org/10.1145/2009916.2009967

Dawn Lawrie, W. Bruce Croft, and Arnold Rosenberg. 2001. Finding Topic Words for Hierarchical Summarization. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '01)*. ACM, New Orleans, Louisiana, USA, 349–357. DOI:http://dx.doi.org/10.1145/383952.384022

Jiyun Luo, Dongyi Guan, and Hui Yang. 2013. InfoLand: Information Lay-of-land for Session Search. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '13)*. ACM, Dublin, Ireland, 1097–1098. DOI:http://dx.doi.org/10.1145/2484028.2484213

Jiyun Luo, Sicong Zhang, and Hui Yang. 2014. Win-win Search: Dual-agent Stochastic Game in Session Search. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '14)*. ACM, Gold Coast, Queensland, Australia, 587–596. DOI:http://dx.doi.org/10.1145/2600428.2609629

Prasanta C. Mahalanobis. 1936. On the Generalised Distance in Statistics. In *Proceedings National Institute of Science, India*, Vol. 2. 49–55. http://ir.isical.ac.in/dspace/handle/1/1268

ODP. 2011. Open Directory Project. (2011). http://www.dmoz.org/

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: Leveraging Generic Patterns for Automatically Harvesting Semantic Relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (ACL-44)*. Association for Computational Linguistics, Sydney, Australia, 113–120. DOI: http://dx.doi.org/10.3115/1220175.1220190

Deepak Ravichandran and Eduard Hovy. 2002. Learning Surface Text Patterns for a Question Answering System. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL '02)*. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, 41–47. DOI: http://dx.doi.org/10.3115/1073083.1073092

Mark Sanderson and Bruce Croft. 1999. Deriving Concept Hierarchies from Text. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '99)*. ACM, Berkeley, California, USA, 206–213. DOI: http://dx.doi.org/10.1145/312624.312679

Mark Sanderson and Dawn Lawrie. 2000. *Building, Testing, and Applying Concept Hierarchies*. Kluwer Academic Publishers. 235–256 pages.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2005. Learning Syntactic Patterns for Automatic Hypernym Discovery. In *Proceedings of the 19th Annual Conference on Neural Information Processing Systems (NIPS 2005)*. Vancouver and Whistler, Canada.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic Taxonomy Induction from Heterogenous Evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (ACL-44)*. Association for Computational Linguistics, Sydney, Australia, 801–808. DOI: http://dx.doi.org/10.3115/1220175.1220276

Emilia Stoica and Marti A. Hearst. 2007. Automating Creation of Hierarchical Faceted Metadata Structures. In *Proceedings of the Human Language Technology Conference (NAACL-HLT)*. Rochester, New York, USA.

Trevor Strohman, Donald Metzler, Howard Turtle, and W. Bruce Croft. 2005. Indri: A Language Model-based Search Engine for Complex Queries. (2005).

Robert Tibshirani, Guenther Walther, and Trevor Hastie. 2000. Estimating the Number of Clusters in a Dataset via the Gap Statistic. In *Technical Report 208, Department of Statistics, Stanford University*.

Xuanhui Wang and ChengXiang Zhai. 2007. Learn from Web Search Logs to Organize Search Results. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*. ACM, Amsterdam, The Netherlands, 87–94. DOI: http://dx.doi.org/10.1145/1277741.1277759

Hui Yang. 2011. *Personalized Concept Hierarchy Construction*. Ph.D. Dissertation. Carnegie Mellon University. http://www.cs.cmu.edu/~huiyang/publication/dissertation.pdf.

Hui Yang. 2014. A Fragment-based Similarity Measure for Concept Hierarchies and Ontologies. In *Proceedings of the 7th International Workshop on Exploiting Semantic Annotations in Information Retrieval (ESAIR '14)*. ACM, Shanghai, China, 41–42. DOI: http://dx.doi.org/10.1145/2663712.2666188

Hui Yang and Jamie Callan. 2009. A Metric-based Framework for Automatic Taxonomy Induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1 (ACL '09)*. Association for Computational Linguistics, Suntec, Singapore, 271–279. http://dl.acm.org/citation.cfm?id=1687878.1687918

Ka-Ping Yee, Kirsten Swearingen, Kevin Li, and Marti Hearst. 2003. Faceted Metadata for Image Search and Browsing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03)*. ACM, Ft. Lauderdale, Florida, USA, 401–408. DOI: http://dx.doi.org/10.1145/642611.642681

Chengxiang Zhai and John Lafferty. 2004. A Study of Smoothing Methods for Language Models Applied to Information Retrieval. *ACM Trans. Inf. Syst.* 22, 2 (April 2004), 179–214. DOI: http://dx.doi.org/10.1145/984321.984322

Sicong Zhang, Dongyi Guan, and Hui Yang. 2013. Query Change As Relevance Feedback in Session Search. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '13)*. ACM, Dublin, Ireland, 821–824. DOI: http://dx.doi.org/10.1145/2484028.2484171