Automatically Labeling Hierarchical Clusters

Pucktada Treeratpituk
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
puck@cs.cmu.edu

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

ABSTRACT
Government agencies must often quickly organize and analyze large amounts of textual information, for example comments received as part of notice and comment rulemaking. Hierarchical organization is popular because it represents information at different levels of detail and is convenient for interactive browsing. Good hierarchical clustering algorithms are available, but there are few good solutions for automatically labeling the nodes in a cluster hierarchy.

This paper presents a simple algorithm that automatically assigns labels to hierarchical clusters. The algorithm evaluates candidate labels using information from the cluster, the parent cluster, and corpus statistics. A trainable threshold enables the algorithm to assign just a few high-quality labels to each cluster. Experiments with Open Directory Project (ODP) hierarchies indicate that the algorithm creates cluster labels that are similar to labels created by ODP editors.

Categories and Subject Descriptors
H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Linguistic processing.*

General Terms
Algorithms, Performance, Experimentation.

Keywords
Document hierarchy, cluster labeling.

1. INTRODUCTION
Government agencies, like most modern organizations, must often quickly organize and analyze large amounts of textual information. Our research is motivated by notice and comment rulemaking, in which U.S. regulatory agencies are required to consider comments submitted by the general public [11]; when an agency receives hundreds of thousands of unique comments and edited form letters during a short period of time, quickly organizing them and identifying the issues raised is a significant challenge. However, the problem of organizing large amounts of text for rapid analysis subsumes notice and comment rulemaking. The U.S. National Archives and Records Administration (NARA) and other government agencies face similar problems. Search engines such as FirstGov[1] have the problem on a smaller scale – given a set of texts, how to quickly organize them and describe their contents.

Hierarchical organization is popular because it represents information at different levels of detail and is convenient for interactive browsing (e.g., Yahoo! [17], the Open Directory Project [13]). At the top of the hierarchy, the collection is organized into a few general categories; as a person descends the hierarchy, she gets greater detail about increasingly specific categories. Typically each document cluster is assigned a descriptor that describes the documents it contains. One goal of hierarchical clustering is to improve the users' ability to browse the collection, so it is very important that the hierarchy has good cluster descriptors. These descriptors can be either category labels, as in Yahoo! Directories, or lists of topical terms.

There is considerable prior research on hierarchical clustering algorithms and their applications in information retrieval and data mining research. However, less attention has been paid to creating good cluster descriptors. Cluster descriptors created automatically often either fail to provide a comprehensive description of the cluster, or consist of lists of terms from which a person must infer a general description.

This paper proposes a simple algorithm that automatically assigns concise labels to hierarchical clusters. The algorithm combines statistical features of the cluster, the parent cluster, and the corpus into a descriptive score. The algorithm is based on the hypothesis that by comparing the word distribution from different parts of the hierarchy, it should be possible to assign appropriate labels to each cluster in the hierarchy.

The rest of this paper is organized as follows. Section 2, describes the hierarchical clustering and hierarchical clustering labeling task. It also discusses the characteristic of a good label for hierarchical clusters. Section 3 presents previous research on cluster labeling and related tasks. Section 4, describes the proposed labeling algorithm. Section 5 presents experimental results. Section 6 summarizes our finding and offers suggestions about possible future improvements.

dg.o2006, May 21–24, 2006, San Diego, California, USA.

[1] http://www.firstgov.gov/

2. LABELING HIERARCHICAL CLUSTERS

Hierarchical clustering partitions a document collection into a small number of clusters, and each cluster is further partitioned into subclusters in a recursive manner. Hierarchical clusters can be constructed by *agglomerative* methods that start with each document in its own cluster and then repeatedly group similar clusters into broader clusters; or by *divisive* methods that start with all documents in one cluster and then repeatedly divide each cluster into more detailed subclusters.

In labeling hierarchical clusters, one assumes the existence of a hierarchy of document clusters. The task is to assign a good descriptor to each cluster node in the hierarchy. The most common cluster descriptors are either concise labels, or lists of terms and phrases. For example, a cluster of documents about natural language processing might be described by the label "natural language processing" or the list of terms "tag, text, linguist, lexicon, corpus, tagger, word, syntax, grammar." A list of terms is often less useful than a single category label, because it requires the user to infer the concept implied by the terms. However, a list of terms is the most common choice for labeling clusters automatically because it fails gracefully; a person can often infer the general description even when a few of the selected terms are poor choices.

Our goal is an algorithm that selects concise cluster labels that are similar to what a person might create manually. A good descriptor for a cluster should not only indicate the main concept of the cluster, but also differentiate the cluster from its siblings and its parent cluster. Consider a cluster of "neural network" documents under a broader "AI" cluster. The parent cluster "AI" may also have "machine learning" and "fuzzy logic" child clusters. In another context, a label "computer science" might be an acceptable descriptor for the "neural network" cluster. However, in the context of this hierarchy, "computer science" does not distinguish the "neural network" cluster from its siblings. The descriptors appropriate for a given set of documents will differ under different hierarchies.

We define the labeling task as follows: Given a cluster of documents in a cluster hierarchy, the goal is to produce appropriate category labels for the cluster. The algorithm is allowed to return a list of plausible labels, ranked by its confidence about how descriptive each label is; the list should be as short as possible. A ranked list of labels is a compromise between a single category label and a list of topical terms.

3. RELATED WORK

Although there has been much research in hierarchical clustering of documents, little focused on labeling the resulting clusters of documents [2][3][4][5][10]. Clustering algorithms can naively label their clusters with the most frequent words in the clusters [2]. Those resulting labels tend to be general, and usually are not good discriminative descriptors. The algorithm may choose to represent its clusters of documents with the documents near each cluster's centroid. One example of algorithms that use this representation is Scatter/Gather [3], which represents a cluster with a list of documents near the cluster's centroid and a list of topical terms. The topical terms are the terms with the highest weights in the cluster centroid. In one report the algorithm showed approximately the top 10 topical terms to the users. In addition to the shortcomings, mentioned above, of using lists of terms as descriptors, this approach does not take into account the hierarchical structure of the cluster hierarchy. The resulting descriptor might be descriptive, but might not discriminate the cluster from its parent or sibling clusters.

There have been attempts to identify cluster labels from word distribution in the hierarchy. Popescul et al. [10], and Glover et al. [4] proposed statistical methods in selecting cluster descriptors, based on the context of the surrounding clusters (parent cluster and sibling clusters). Popescul proposed to use the statistical test χ^2 to detect difference in word distribution across the hierarchy. At each cluster node in the hierarchy, starting from the root, the χ^2 test is used to detect a set of words that is equally likely to occur in any of the subclusters of the current node. Those words are considered to be non-descriptive terms for every subcluster of the current node, and thus are removed from every subcluster. After the χ^2 test is used to removed non-descriptive words from every cluster node, the algorithm labels each cluster with the list of the remaining words at that cluster node ranked by the word frequency.

Glover et al. [4] showed how a simple model based only on terms' document frequency statistic can be used to select parent, child and self descriptors for document clusters, especially for web pages. The label candidates were extracted from the web page's content, anchor texts and extended anchor texts. Anchor texts of a web page are the hyperlinked words that link to the web page. Extended anchor texts refer to the words that occur before and after the anchor texts including the anchor texts themselves. Labels were selected and ranked using document frequency and some preset cutoff values. In their experiment, they found that labels extracted from anchor texts and extended anchor texts provide better description than ones extracted from the page's content. This is because web pages often do not contain words that describe their categories. However, obtaining anchor texts and extended anchor texts is an expensive operation, and requires one to know the hyperlink structure of the World Wide Web. Furthermore, the hyperlink structure is generally not available in non-webpage document collection.

Another research area closely related to cluster labeling is automatic ontology construction. It should be noted that while a ontology hierarchy has well-defined parent-child relationship, such as hypernyms-hyponym and meronymy (part-whole), a document hierarchy of the same collection does not necessary have to reflect the same parent-child relationship. The higher flexibility in hierarchal structure of a document hierarchy might better serve for task such as browsing.

There have been some works on creating ontology hierarchies using clustering based techniques. These approaches require that the resulting hierarchy be automatically labeled. Caraballo [1] constructed a noun hierarchy of hypernyms automatically from text. The noun hierarchy is constructed using bottom-up clustering approach, grouping nouns based on conjunction and apposition. In order to label each internal cluster, a set of possible hypernyms of every noun in the cluster is extracted from the text using a linguistic pattern. The noun that has the largest number of hyponym relations with the noun in the clusters is assigned as the cluster label.

Pantel et al. [9] automatically assigned label to semantic classes, generated from their clustering algorithm. For each semantic class, a subset of concepts in the class that is most likely to represent the semantic class is selected as class representatives. These representative concepts are then used to extract label candidates using some lexical patterns. The label candidate with the highest mutual information with the class representatives is assigned as class label.

4. ALGORITHMS

Glover et al. [4] showed that a simple term frequency analysis could predict the labels of document clusters. Their algorithm is based on the hypothesis that a word that is very common in the cluster, but relatively rare in the collection, is likely to be a good cluster descriptor. They selected cluster descriptor candidates based on the following criteria:

$$\text{Candidates} = \{\text{phrase } p \mid DF_C / |C| < \text{maxColPos and } DF_S / |S| > \text{minSelfPos}\}$$

where DF_C is the number of documents in the collection that contains the phrase p ("document frequency"), and DF_S is the number of documents in the cluster (the "self cluster" S) that contain the phrase p. $|C|$ and $|S|$ denote the number of documents in the collection and in the self cluster. maxColPos and minSelfPos are thresholds. Phrases that appear more than minSelfPos times per document, on average, in "self cluster" documents, and less than maxColPos times per document, on average, in the collection are considered to be the in the label candidate set. Phrases in the candidate set are ranked according to their DF_S values. Every phrase in the collection is considered, without stemming or stopwords removal [4].

There are several limitations to Glover's threshold-based method. First, the performance of the algorithm is sensitive to preset threshold values (maxColPos, minSelfPos), and the optimal thresholds vary between clusters. Second, multiword phrases normally have lower $DF_S / |S|$ values than single words, thus are rarely selected as a descriptors. Third, documents often do not contain words that describe their categories, so basing the decision mainly on $DF_S / |S|$ generally does not work well.

Due to these limitations, we propose a more general labeling algorithm that allows us to incorporate more features in selecting the cluster descriptors.

Labeling Algorithm:

First, we assume that the algorithm has access to a general collection of documents E, representing the word distribution in general English. This English corpus is used primarily in selecting label candidates, as explained below.

Given a cluster S and its parent cluster P, which includes all of the documents in S and in the sibling clusters of S, the algorithm selects labels for the cluster S with the following steps:

1) **Collect phrase statistics:** For every unigram, bi-gram, and tri-gram phrase p occurring in the cluster S, calculate the document frequency and term frequency statistics for the cluster, the parent cluster and the general English corpus.
2) **Select label candidates:** Select the label candidates from unigram, bi-gram, and tri-gram phrases based on document frequency in the cluster and in general English language.
3) **Calculate the descriptive score:** Calculate the descriptive score (*DScore*) for each label candidate, then sort the label candidates by these scores.
4) **Calculate the cutoff point:** Decide how many label candidate to display based on the descriptive scores.

Each step is described in more detail, below.

4.1 Collecting Phrase Statistics

For each phrase appearing in the cluster, collect the following statistics: document frequency (*DF*), and term frequency (*TF*) with respect to the cluster S, the parent cluster P and the general English corpus E. Document frequency of a phrase p with respect to a cluster C, denoted by DF_C, is the number of documents in the cluster that contain p. Term frequency of a phrase p in a cluster C, denoted by TF_C, is total number of occurrences of p in the cluster.

4.2 Select Label Candidates

Instead of considering every phrase occurring in the cluster, we hypothesize that, although a good descriptor need not occur in the majority of the documents in the cluster, it should occur in at least 20% of the documents in the cluster. Since phrases in general occur less frequent than single words, the selection criteria are slightly different in the case of bigrams and trigrams: The algorithm only considers bigram and trigram phrases that occur in at least 5% of the documents in the cluster. These cutoffs improve the efficiency of the algorithm. Low frequency phrases usually have low weights, thus these thresholds generally don't prune phrases that would be ranked highly otherwise.

Common words (stopwords) are also removed from consideration. The algorithm considers any words that occur in more than 20% of the general English corpus to be stopwords. In the case of the non-unigram phrases, the algorithm considers any phrases that contain only words that occur in more than 30% of the documents in the general English corpus E to be stopwords. This cutoff was chosen conservatively by analyzing the word distribution in general English corpora, trying not to exclude descriptive words.

4.3 Descriptive Score (DScore)

The descriptiveness of a label with respect to a cluster is measured by the descriptive score (DScore). For a phrase p, the descriptive score is based on the features described below.

Normalized Document Frequency ($DFc / |C|$)

Normalized document frequency is the fraction of the cluster that contains the phrase p.

$$\textit{normalized } DF_C = \frac{DF_C}{|C|}$$

In general a label candidate that occurs in more documents in the cluster is expected to be a better descriptor than one that rarely occurs. The algorithm computes *normalized DF* for both the self cluster S and the parent cluster P ($DF_S / |S|$, $DF_P / |P|$). A good descriptor should occur relatively frequent in the parent cluster, but occur very frequent in the self cluster.

TFIDF

This is similar to a traditional *TFIDF* value used in information retrieval.

$$TFIDF_C = TF_C * \log(\frac{|C|}{DF_C})$$

As in traditional IR, a phrase with high *TFIDF* value is expected to be important to the cluster, thus possibly is also a good cluster descriptor. The *TFIDF* score favors phrases that appear multiple times per document. The algorithm computes *TFIDF* for both the self cluster and the parent cluster ($TFIDF_S$, $TFIDF_P$).

Rank *of TFIDF*, and *nDF* (*r(TFIDF), r(normalized DF)*)

Four *rankings* are computed for every label candidate based on the features DF_S / |*S*|, DF_P / |*P*|, $TFIDF_S$, and $TFIDF_P$. For example, for DF_S / |*S*|, the algorithm sorts every label candidate according to its DF_S / |*S*| score; the label with the highest value is assigned rank 1, denoted by r(DF_S / |*S*|) = 1. The label with the second highest score is assigned rank 2, and so on. Tied scores produce tied rankings. A good descriptor is expected to have a relatively high rank of DF_P / |*P*| and even higher rank of DF_S / |*S*|.

Rank features, e.g. r(DF_S / |*S*|), convey similar information as their quantitative counterparts, e.g. DF_S. However, rank features may be less sensitive than normalized *DF* and *TFIDF* values, and thus may be more comparable across categories.

Boost in Ranking

Since we hypothesize that a good descriptor probably has a relatively high rank of normalized DF_P (relatively frequent in the parent cluster), and even higher rank of normalized DF_S (very frequent in the self cluster), we measure this boost in ranking of *nDF* with the following measure:

$$\log\left[r\left(\frac{DF_P}{|P|}\right)\right] - \log\left[r\left(\frac{DF_S}{|S|}\right)\right]$$

The algorithm computes the boost in ranking in log-scale because the change in ranking is more significant at the top of the ranking (those which have high document frequency). For example, a label that moves from being the 200th most frequent phrase in the parent cluster to the 100th most frequent phrase in the self cluster, is probably less descriptive than another label that moves from the 100th most frequent phrase in the parent cluster to the 5th most frequent phrase in the self cluster.

In addition to the boost in ranking of normalized document frequency, the algorithm also computes the boost in ranking of *TFIDF*, with the following formula:

$$\log[r(TFIDF_P)] - \log[r(TFIDF_S)]$$

If *TFIDF* is related to the topicality of the phrase, then a good descriptor is expected to have a higher *TFIDF* rank in the self cluster, compared to its *TFIDF* rank in the parent cluster. The algorithm also computes the boost in ranking of *TFIDF* in log-scale for the same reason as in the case of normalized *DF*.

Phrase Length (*LEN*)

The phrase length is the number of terms in the phrase. While the document frequency feature prefers a shorter phrase to a longer phrase, LEN prefers the longer phrases to shorter ones.

The algorithm combines every feature into one descriptive score with a linear model. Thus the algorithm computes how descriptive a phrase *p* is as a label for the cluster S, with parent cluster P with the following formula:

$$\begin{aligned} DScore_p &= c_0 + c_1 * LEN \\ &+ c_2 * DF_S/|S| + c_3 * DF_P/|P| \\ &+ c_4 * TFIDF_S + c_5 * TFIDF_P \\ &+ c_6 * r(DF_S/|S|) + c_7 * r(DF_P/|P|) \\ &+ c_8 * r(TFIDF_S) + c_9 * r(TFIDF_P) \\ &+ c_{10} * [\log(r(DF_P/|P|)) - \log(r(DF_S/|S|))] \\ &+ c_{11} * [\log(r(TFIDF_P)) - \log(r(TFIDF_S))] \end{aligned}$$

Each label candidate is sorted by its descriptive score.

The weights of each feature are estimated using linear regression and training data. Linear regression attempts to estimate the expected value of a variable Y given the values of a set of features Xi, by assuming a linear relationship between Y and X_i. Thus, Y can be expressed as linear combination of features Xi:

$$Y = b_0 + \sum b_i X_i + e$$

where e is a random variable residue (error term), with mean zero. The coefficients b_i for all i are optimized so that the sum of the residue square in the training data is minimized.

In order to train the linear regression model, since the correct descriptive score is not known for each label candidate, we have to estimate the descriptive score of a label candidate. We estimate each label candidate's descriptive score based on how much the label overlaps with the correct category label in a set of training data. We define the DScore estimate of a label *L*, with respect to the correct label *CL* as:

$$DScore_L^* = \max_{SL \in Synonym(L)} \{\frac{overlap(SL,CL)}{\max\{len(SL), len(CL)\}}\}$$

where overlap(*SL*,*CL*) is the number of terms that are shared between *SL* and *CL*, and len(*X*) denotes the length of *X*. If the label candidate or a synonym of the label candidate is the same as the correct category label, then the DScore estimate is 1. The estimation of DScore that we use to train the linear regression model is only a heuristic value, because many good descriptors would have the DScore estimates of zero, since they do not overlap with the correct label.

4.4 Cutoff Model

By default, the algorithm displays the 5 labels with the highest descriptive scores as the cluster descriptor. However, we observe that even in a short list of five labels there is generally a big drop-off in the descriptive scores at some point. The big drop-off in the descriptive scores often separates good labels from bad labels. The algorithm can use this information to decide how many labels to display. If the top-ranked label has a very high descriptive score compared to the rest of the label candidates then the algorithm can be very confident that the top-ranked label is the correct descriptor, and thus display only the top-ranked label. On the other hand, if all label candidates have

similarly low DScores (only small gaps between each consecutive label), there is less certainty about which labels are best, so more labels are displayed.

The following linear model is used to decide how many labels to display.

$$\begin{aligned} \#Displayed = c_0 &- c_1 * (DScore_{L1} - DScore_{L2}) \\ &- c_2 * (DScore_{L2} - DScore_{L3}) \\ &- c_3 * (DScore_{L3} - DScore_{L4}) \\ &- c_4 * (DScore_{L4} - DScore_{L5}) \end{aligned}$$

The weights in the model are optimized using linear regression and training data. We expect the optimized C_0 to be around 5, while expecting the weights C_1, C_2, C_3, C_4 to be in increasing order.

The same training data used to train the descriptive score model in Section 4.3 can be used to generate training instances for the cutoff model. The cutoff model is trained based on the top-ranked labels produced by the trained descriptive model. For each category in the training data, the 5 labels with the highest predicted descriptive score are determined. The optimal number of labels to display is defined as the rank of the label (from 1 to 5) that has the maximum overlap with the correct category label. If there is a tie, then the label furthest down the list is picked. Thus the cutoff model is trained based on the top-5 predicted descriptive scores from each training category.

5. EXPERIMENTAL RESULTS

A set of experiments was conducted to evaluate the effectiveness of the algorithm at selecting labels. We describe the data and evaluation measures first, followed by descriptions of the experiments and their results.

5.1 Data Collections

The Open Directory Project[2] was used as a source of documents (web pages), hierarchical organization, and "ground truth" labels assigned by human editors. We randomly sampled 20,462 web pages from the ODP hierarchy to use as background model representing general English (collection statistics). We separately sampled another subset of ODP hierarchy to use as our training and testing ground truth data.

We selected total of 165 subcategories from ODP under 9 categories.

- Computers / artificial intelligence.
- Computers / security.
- Health / alternative.
- Health / medicine.
- Health / medicine / imaging.
- Health / medicine / surgery.
- Health / conditions and diseases.
- Health / conditions and diseases / digestive disorders.
- Business / management.

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

Every subcategory from the 9 parent categories was included, with the exception of alias subcategories (because those subcategories mainly belong somewhere else in the hierarchy), even ones with common words as labels such as Surgery / General. In total, the constructed hierarchy contains 25,143 web pages. The selected subcategories vary both in depth (between level two and level three with respect to the Open Directory root) and in number of web pages in each cluster. Figure 1 shows a partial snapshot the document hierarchy.

Business (3,318)
- Management (3,318)
  - Benchmarking and Best Practices (35)
  - Business Process Analysis (41)
  - …

Computers (4,982)
- Artificial Intelligence (1,616)
  - Academic Departments (60)
  - Agents (98)
  - …
- Security (3,366)
  - Advisories and Patches (71)
  - FAQs, Help, and Tutorials (11)
  - …

Health (16,843)
- Alternative (5,387)
  - Acupuncture and Chinese Medicine (502)
  - Alexander Technique (129)
  - …
- Conditions and Diseases (8,204)
  - Allergies (169)
  - Blood Disorders (176)
  - …
- Medicine (3,252)
  - Imaging (173)
    - Computer Tomography Scanning (11)
    - Magnetic Resonance Imaging (22)
  - Research (107)
  - …
  - Surgery (506)
    - Cardiothoracic (112)
    - Cosmetic and Plastics (73)
    - General (86)
    - Head and Neck (4)

Figure 1. shows subset of the ODP hierarchy used as ground truth data, number of documents in each cluster is showed in the parentheses

5.2 Evaluation Measure

We define the cluster-labeling task as a descriptor-ranking problem. In the evaluation we need to specify our criteria in assessing the quality of the ranking produced. In comparing our labels to the correct ODP label, we use the following two definitions of a correct label: Exact match and partial match.

5.2.1 Definitions of a correct label

For a given category with self-label *S*, and parent-label *P*:

Exact Match: A label *L* is an exact match of the correct label *S* if there exists a synonym *SL* of *L* such that *SL* is equal to "*S*", "*S P*" or "*P S*." For example, for the category "medical /

research," labels such as "research," "medical research," and "research medical" would be classified as exact match labels.

Partial Match: A label *L* is a partial match of the correct label S if there exists a synonym *SL* of *L* such that *SL* shares a term with "*S*", "*S P*" or "*P S*." For example, for the category "management / business process analysis," labels such as "business," "process," "business management," "management analysis" would be classified as partial matches.

The synonym list for each word was obtained automatically from WordNet [16]. For both of these definitions of a correct label, we compute the following evaluation measures.

5.2.2 Match at top N results (Match@N)

Match@N indicates whether the top N results contain any correct labels. It is a binary indicator, and monotonically increases as N increases.

5.2.3 Precision at top N results (P@N)

Precision is computed as the number of labels in the top N results that match the correct categories label divided by N. P@N measures the percentage of correct answers that are displayed in ranks 1-N. In general, low precision is undesirable.

5.2.4 Mean Reciprocal Rank (MRR)

Mean reciprocal rank is the mean of the reciprocal of the rank of the first correct label. If the first correct label is ranked as the 3rd label, then the reciprocal rank (RR) is 1/3. If none of the first N responses contains a correct label, RR is 0. RR is 1 if the highest ranked label matches the correct label.

5.2.5 Mean Total Reciprocal Rank (MTRR)

Sometimes there is more than one aspect to a category; for example, the category "acupuncture and Chinese medicine" has two correct aspects, "acupuncture" and "Chinese medicine." MTRR is similar to MRR, however, instead of considering only the rank of the first correct label as in MRR, MTRR takes into account all correct labels. Of the algorithm ranks "acupuncture" and "Chinese medicine" as the 2nd and the 4th labels, then the TRR (total reciprocal rank) is ½ + ¼ = ¾ while RR = ½.

Our evaluation methodology is extremely strict because it

ODP category labels [parent-label/self-label]	#Docs	Labels
artificial intelligence/agents	84	agent, software agent
artificial intelligence/conferences and events	72	conference, artificial intelligence, international conference
artificial intelligence/genetic programming	65	genetic, genetic algorithm
artificial intelligence/philosophy	46	philosophy, mind, science
artificial intelligence/vision	61	vision, compute vision
security/conferences	14	secure conference, conference attend
security/honeypots and honeynets	62	honeypot, attack
security/news and media	73	attack, vulnerable, hack
alternative/apitherapy	18	bee, honey
alternative/ear candling	10	ear candle, ear
alternative/herbs	258	herb, plant
alternative/iridology	18	iridology, iris
alternative/non-toxic living	53	toxic, environmental, safe
alternative/reflexology	95	reflexology, reflexologist
alternative/urine therapy	6	urine
conditions and diseases/chronic illness	53	ill, chronic ill, chronic
digestive disorders/esophagus	35	reflux, heartburn
digestive disorders/pancreas	10	pancreas, pancreatiti
conditions and diseases/food and water borne	75	food, diarrhea
conditions and diseases/musculoskeletal disorders	473	pain, arthritis, joint
conditions and diseases/skin disorders	383	skin, treat
medicine/education	437	continue medical educate, medical educate, medical school
imaging/computer tomography scanning	11	tomography, compute tomography, compute
imaging/x-ray	15	breast cancer, cancer
medicine/reference	119	database, medical subject head, library
surgery/cryosurgery	7	cryosurgery, treat, cryotherapy
surgery/orthopedics	19	orthopaedic, hip
surgery/transplant	36	transplant, transplantation
management/business process analysis	31	business process, business process model
management/management science	430	university, research, paper
management/value based management	8	shareholder value, consult firm, firm

Figure 2. clusters' labels predicted by the descriptive score with cut-off model for 31 categories.

measures agreement with the single ODP category label selected by the human editor, whereas in fact there might be several equally good category labels. For example, in the category "cardiovascular disorder," our algorithm might select "heart" and "heart disease" as labels for the cluster, which would be acceptable labels to most human assessors. Our automatic evaluation would judge "heart" and "heart disease" as unacceptable answers. We alleviate some of this problem by accepting synonymous labels, as defined by WordNet synonym lists [16] in our evaluation. However, there are still cases such as "cardiovascular" and "heart", which are not actually synonyms, but which most people would consider acceptable substitution labels in the context of "heart disease."

5.3 Experimental Setup & Results

We evaluated our model on the ground truth ODP data of 165 categories, with a total of 25,143 web pages. Each web page was parsed and all HTML tags, images, and JavaScript were removed in the preprocessing step. Each term was stemmed using Krovetz's stemmer [6]. No stopwords list was used, because we expected the algorithm to be able to distinguish the collection-specific stopwords from the content words. The goals of our experiments were three-fold. First, we wanted to evaluate the quality of the cluster labels produced by the algorithm, in comparison with the previous technique. Second, we wanted to investigate the performance of the model that uses only rank features. Third, we wanted to investigate how the model performs if the hierarchical cluster is noisy, as would be the case when using a hierarchical clustering algorithm to organize documents.

5.3.1 Performance Comparison

Glover's threshold-based algorithm was used as the baseline system. The experiment used five-fold cross-validation; in each fold, the training data was used to estimate the optimal parameters for each algorithm. In the case of the threshold-based model, the training data was used to find the optimal threshold values. In our algorithm, the training data was used to learn the weights in the linear model of the descriptive score. This training data was also used to train the cutoff model to predict how many labels to show.

In the training phase we first generated training instance-value pairs for the descriptive score and the feature set training. For each category in the training data, we estimated the DScore for each of its label candidates as described in Section 4.3. We also trained the cutoff model as described in Section 4.4.

The experiment was run on the baseline system and two versions of our algorithm: One with just the descriptive score, and another with both the descriptive score and the cutoff model. Tables 1, 2 and 3 show results for the three algorithms.

Table 1. Match@N with exact, and partial match criteria.

Match@N (exact)	N = 1	N = 2	N = 3	N = 4	N=5
Glover's	0.27	0.35	0.42	0.46	0.50
DScore	0.36	0.50	0.58	0.62	0.64
DScore + Cutoff	0.37	0.49	0.55	0.55	0.55
Match@N (partial)					
Glover's	0.39	0.52	0.60	0.64	0.68
DScore	0.53	0.63	0.69	0.72	0.76
DScore + Cutoff	0.52	0.63	0.66	0.66	0.66

Table 2. Precision@N with exact, and partial match criteria.

P@N (exact)	N = 1	N = 2	N = 3	N = 4	N=5
Glover's	0.27	0.18	0.16	0.13	0.12
DScore	0.36	0.27	0.22	0.19	0.17
DScore + Cutoff	0.37	0.28	0.27	0.26	0.26
P@N (partial)					
Glover's	0.39	0.32	0.30	0.28	0.25
DScore	0.53	0.45	0.40	0.38	0.35
DScore + Cutoff	0.52	0.46	0.43	0.43	0.43

Table 3. MRR, MTRR, and Average Length statistics.

Exact	MRR	MTRR	Avg. Length
Glover's	0.35	0.38	5
DScore	0.47	0.53	5
DScore + Cutoff	0.45	0.47	2.6
Partial			
Glover's	0.50	0.68	5
DScore	0.61	0.94	5
DScore + Cutoff	0.59	0.74	2.6

Both descriptive score models outperform the threshold-based approach. The Match@1 values are around 0.36 in exact match and 0.53 in partial match for both descriptive score models, compared to 0.27 and 0.39 for baseline model. This means that in almost half the categories, the descriptive score predicts the correct label with the top rank label. The precision of both descriptive score models is higher than the baseline model. This suggests that the lists of labels produced by our descriptive score contain more good labels than the ones produced by the baseline. This is also supported by the higher MTRR measure for the descriptive score model.

The average number of labels displayed with the cutoff model is 2.6. By choosing to display fewer labels, the algorithm with the cutoff model has a lower number of correct matches (M@N) and also lower MRR, and MTRR. However, the precision of the list of labels produced is higher, because the model tries not to show low-quality labels. We believe that the tradeoff in lower MRR with higher precision is worthwhile because a shorter list of labels makes it easier for users to understand the content of the cluster. However, a user study would be needed to verify our conjecture.

Figure 2. shows the labels produced by the DScore+Cutoff model along with the corresponding ("correct") ODP labels. In most categories, the labels produced by the model match the category labels in the ODP. Even when model did not produce exactly the same labels as the ODP, the labels assigned by the model provide a similar description. For example, in the category, security / news and media, the list of labels, "attack, vulnerable, and hack" describes what most of the documents discuss.

The algorithm works well in spite of a very heuristic method used to generate scores for ODP labels during training. We believe that this effectiveness is because the trained regression model does not need to predict an exact DScore; it needs only to produce a relative score for each label that is suitable for ranking them. One thing to note is that while the algorithm ranks labels using the relative importance of terms between the parent cluster and the self cluster, it does not use information about sibling clusters. The algorithm could potentially rank the same labels highly for multiple sibling clusters. However, in our evaluation with the ODP data, this was rarely the case. All sibling clusters are pooled together to form the parent cluster, so if the hierarchy is well-formed such that every sibling cluster is of roughly the same granularity, the highly ranked terms in the parent cluster are similar to the highly ranked terms of its children, yielding small relative differences. Comparing a child to its parent cluster has an indirect effect similar to comparing against its siblings. We suspect that in a less well-formed hierarchy the algorithm would need to consider information about each individual sibling in order to assign discriminative labels.

5.3.2 Using Only Rank Features

To test the hypothesis that one can identify a good label for a cluster based only on rank features, the descriptive score formula in Section 4.1 was modified to use only the rank features and the boost in ranking. Tables 4, 5, and 6 show the results.

Table 4. Match@N with exact, and partial match criteria for rank-features model.

Match@N (exact)	N = 1	N = 2	N = 3	N = 4	N=5
Glover's	0.27	0.35	0.42	0.46	0.50
DScore	0.35	0.52	0.57	0.59	0.64
DScore + Cutoff	0.35	0.52	0.55	0.55	0.55
Match@N (partial)					
Glover's	0.39	0.52	0.60	0.64	0.68
DScore	0.53	0.64	0.72	0.73	0.76
DScore + Cutoff	0.53	0.63	0.68	0.69	0.69

Table 5. Precision@N with exact, and partial match criteria for rank-features model.

P@N (exact)	N = 1	N = 2	N = 3	N = 4	N=5
Glover's	0.27	0.18	0.16	0.13	0.12
DScore	0.35	0.28	0.22	0.19	0.16
DScore + Cutoff	0.35	0.29	0.27	0.27	0.27
P@N (partial)					
Glover's	0.39	0.32	0.30	0.28	0.25
DScore	0.53	0.45	0.41	0.37	0.35
DScore + Cutoff	0.53	0.45	0.44	0.44	0.44

Table 6. MRR, MTRR, and Average Length statistics for rank-features model.

Exact	MRR	MTRR	Avg. Length
Glover's	0.35	0.38	5
DScore	0.47	0.53	5
DScore + Cutoff	0.44	0.48	2.5
Partial			
Glover's	0.50	0.68	5
DScore	0.62	0.94	5
DScore + Cutoff	0.60	0.77	2.5

The learned descriptive model based only on rank features is as followed:

$$\begin{aligned} DScore(p) = {} & 0.122 \\ & +0.0000 * r(DF_S / \#S) \\ & -0.0001 * r(DF_P / \#P) \\ & +0.0000 * r(TFIDF_S) \\ & -0.0001 * r(TFIDF_P) \\ & +0.0509 * [\log(r(DF_P / \#P)) - \log(r(DF_S / \#S))] \\ & +0.1874 * [\log(r(TFIDF_P)) - \log(r(TFIDF_S))] \end{aligned}$$

The model performs surprisingly well considering that it uses only ranking features. Its MRR is 0.47 for exact match and 0.62 for partial match definition, which are at the same level comparing to the models that use all features.

5.3.3 Noise Resistance

So far we have assumed that the document hierarchy given to the algorithm correctly clusters every document with the same concept together. However, this is rarely the case, because the hierarchical clusters that need automatic labeling are usually produced by imperfect hierarchical clustering algorithms. To evaluate how the algorithm performs in a more realistic setting, another experiment was conducted with noise introduced into our ground truth data.

Consider a cluster *P* that has the set of subclusters, denoted as *children(P)*. For each document in any subcluster of *P*, the document is reassigned to another subcluster of P with a probability *N* (Noise %); with the probability *1-N*, the document remains in the correct subcluster. The probability that a document is reassigned to a subcluster *C* of *P* is proportional to the size of the cluster *C*. So the probability that a document *d* in

a the cluster P is assigned to a subcluster C of P, denoted by $Pr(assigned(d, C))$, is:

$$\Pr(assigned(d,C)) = \begin{cases} 1-N, \text{if } d \in C \\ N * \dfrac{|C|}{\sum_{R \in children(P) \text{ and } d \notin R} |R|}, \text{if } d \notin C \end{cases}$$

where $|C|$ denotes the number of documents that originally belonged to the cluster C.

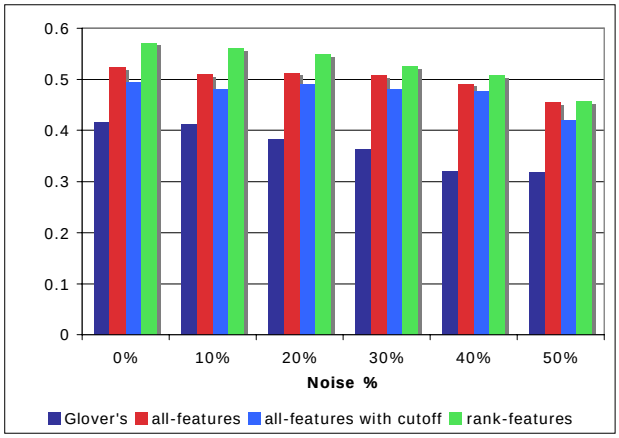

Figure 3. shows MRR for different noise probability levels

Figure 3 shows the performance comparison on exact match definition between different algorithms at noise levels from 0% to 50% on 94 categories of the OPD ground truth data. At 0% noise level, every document is correctly assigned. At 50% noise level, each cluster has approximately 50% of its documents correctly assigned, and the rest are documents that should be in its sibling clusters. We didn't investigate noise levels higher than 50% because the cluster identity is no longer coherent when most of the cluster is assigned incorrectly. In Figure 3, *Glover's* refers the baseline algorithm described in Section 5.3.1; *all-features* and *all-features with cutoff* refer to the models that calculate DScore based on every feature described in Section 4.3, with and without the cutoff model respectively; and *rank-features* refers to the model described in Section 5.3.2, which only used rank-related features to calculate DScore.

As expected the performance of every algorithm decreases as more noise is introduced. At 10% to 30% noise level, there is almost no change in the performance of any model. The MRR of every model drops around 0.1 with 50% noise. However, even with 40% noise level, our algorithms still perform at around 0.5 MRR, which means that on average there is a good descriptor in the top two labels. The decreases in performance mostly came from the small categories, which are more easily disrupted by the introduction of a few documents. In general, our algorithm was not sensitive to random noise.

One might argue that this result is not surprising, because randomly assigning documents to sibling clusters does not change the underlying distribution of words in each cluster. A more realistic simulation would assign a document to sibling clusters based on the similarities between the document and clusters' centroids. Such a simulation would better reflect the errors normally produced by a clustering algorithm. Although a further study is needed to assess the noise tolerance of the model under this scenario, we believe that our experiment shows promising initial result.

6. CONCLUSION AND FUTURE WORK

Tools that automatically organize and assist in the analysis of large amounts of text documents are becoming a requirement in many organizations. There has been considerable research on automatically organizing text documents into hierarchical clusters suitable for interactive browsing, but much less research on how best to automatically describe or label hierarchies to support interactive browsing.

This paper presents a simple trainable algorithm that selects a few 1-3 word labels to describe each cluster in a document hierarchy. The algorithm dynamically decides how many labels to select for each cluster; in our experiments, it average about 2.6 labels per cluster. Experiments using Open Directory Project data demonstrated that the labels produced by the algorithm often match the labels chosen by human editors. Preliminary experiments suggest that the algorithm is also robust with respect to clustering errors, although additional research is required to settle this question.

Our research and most prior research focused on the use of statistical features to select and rank features; a distinguishing feature of our research is the use of statistics from a corpus of general English, the parent cluster, and the cluster to be labeled. However, perhaps more interesting is the discovery that the algorithm can select good descriptors using only rank-based features, and that rank-based features provide more robust results than more detailed numeric features.

Error analysis showed that most of errors come from clusters containing small numbers of documents. The small number of observations in small clusters can make good and bad labels indistinguishable; minor variations in vocabulary can also produce statistical features with spuriously high variance. To improve the performance of the algorithm on small clusters it may be necessary to incorporate lexical features, for example the number of word senses for a candidate label, or positional features sensitive to where terms occur in a document, for example in a title or in a lead sentence. The work described here demonstrates that it is realistic to aim higher than the lists of characteristic terms that have been the norm in prior research on automatic labeling, but it is nonetheless just the first step.

7. ACKNOWLEDGEMENTS

This research was supported by a Thai Ministry of Science, Technology and Environment Scholarship, and by NSF grants EIA-0327979 and IIS-0429102. Any opinions, findings, conclusions, or recommendation expressed in this paper are the authors', and do not necessarily reflect those of the sponsors.

8. REFERENCES

[1] Caraballo, S. Automatic Acquisition of a hypernym-labeled noun hierarchy from text. In Proceedings of the Association for Computational Linguistics Conference, 1999.

[2] Chuang S., and Chien L. A practical web-based approach to generating topic hierarchy for text segments. In

Proceedings of the 20th International Conference on Information and Knowledge Management, 2004.

[3] Cutting D. R., Karger D. R., and Pederson J. O. Constant interaction-time Scatter/Gather browsing of very large document collections. In Proceedings of International ACM Conference on Research and Development in Information Retrieval, 1993.

[4] Glover, E., Pennock, D., Lawrence, S. and Krovetz, R. Inferring hierarchical descriptions. In Proceedings of the 20th International Conference on Information and Knowledge Management, 2002.

[5] Glover, E., Tsioutsiouliklis, K., Lawrence, S., Pennock, D., and Flake, G. Using web structure for classifying and describing web pages. In Proceedings of International Conference on World Wide Web, 2002.

[6] Krovetz, R. Viewing morphology as an inference process. In Proceedings of International ACM Conference on Research and Development in Information Retrieval, 1993.

[7] Lawrie, D., Croft, W. B., and Rosenberg, A. L. Finding topic words for hierarchical summarization. In Proceedings of international ACM conference on research and development in information retrieval, 2001.

[8] Muller, A., Dorre, J. Gerstl, P., and Seiffert, R. The TaxGen framework: automating the generation of a taxonomy for a large document collection. In Proceedings of the 32nd Hawaii International Conference on System Science, 1999.

[9] Pantel, P., and Ravichandran, D. Automatically labeling semantic classes. In Proceedings of the Human Language Technology and North American Chapter of the Association for Computational Linguistics Conference. 2004.

[10] Popescul, A., and Ungar, L. Automatic labeling of document clusters. Unpublished manuscript, available at http://citeseer.nj.nec.com/popescul00automatic.html, 2000.

[11] Yang, H. and Callan, J. Near-duplicate detection for eRulemaking. In Proceedings of the National Conference on Digital Government Research (DG.02005), 2005.

[12] Zeng, H., He, Q., Chen Z., Ma, W., and Ma J. Learning to cluster web search results. In Proceedings of International ACM Conference on Research and Development in Information Retrieval, 2004.

[13] Open Directory Project (ODP).

[14] eRulemaking Testbed. http://hartford.lti.cs.cmu.edu/eRulemaking/Data.html.

[15] Weka, Data Mining Software, University of Waikato.

[16] WordNet, a lexical database for the English language.

[17] Yahoo!