Ontology Generation for Large Email Collections

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

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

ABSTRACT
This paper presents a new approach to identifying concepts expressed in a collection of email messages, and organizing them into an ontology or taxonomy for browsing. It incorporates techniques from text mining, information retrieval, natural language processing and machine learning to generate a concept ontology. Nominal N-gram mining is used to identify candidate concepts. Wordnet and surface text pattern matching are used to identify relationships among the concepts. A supervised clustering algorithm is then used to further cluster the concepts. The experiments show that the approach is effective.

Categories and Subject Descriptors
H.4 [**Information Systems Applications**]: Miscellaneous

Keywords
concept ontology, supervised clustering, eRulemaking

1. INTRODUCTION
Notice and comment rulemaking requires administrative agencies of the U.S. government to issue draft versions of proposed regulations, seek comments from stakeholders and the public, and respond in the final rule to substantive issues raised during the comment period. Each year a few high profile rules attract hundreds of thousands of comments. By law, agencies must consider *every* substantive issue raised during the comment period. Political constraints sometimes require an agency to process comments and issue a final rule quickly. When comment volume is high and the time for processing comments is short, evaluating each comment quickly and thoroughly is a significant burden on the agency.

In recent years, agencies have begun to accept comments in digital form, for example via email or Web portals such as regulations.gov[1]. Digital submission enables new tools to assist agency personnel in organizing and evaluating comments. Recent research developed tools such as duplicate and near-duplicate detection [20, 21] and stakeholder recognition [1]. Although these tools may be useful, manual inspection of each unique comment is still required to determine what issues are raised in the comments.

This paper presents a new tool that automatically identifies the concepts discussed in each comment, and organizes them into an ontology (taxonomy). Browsing hierarchies such as the Yahoo! Directory and the Google Directory are a popular method of quickly discovering the "lay of the land" in a large text corpus. Such a taxonomy of concepts for comments on a proposed rule might help a rule writer more quickly understand the range of issues that were raised, and enable "drilling down" into comments that raise a particular issue.

There is considerable prior research on creating such taxonomies manually, interactively [9, 11, 12, 14, 17, 19], and automatically [2, 3, 4, 5, 6, 7, 13, 15]. The main tasks are recognition of concepts, discovery of relationships among concepts, and organization into an appropriate hierarchy. Creating taxonomies is generally recognized as a difficult task, due to the need to maintain consistent levels of abstraction among siblings and coherent relationships among ancestors and descendents.

This paper develops techniques for automatically building an ontology[2] for large public comments corpora. A set of candidate concepts is created by extracting nominal bigrams and trigrams from the corpus. Several techniques are used to organize the candidate concepts into sibling, ancestor, and descendant relationships. For instance, concept candidates that share the same head nouns or whose head nouns are in the same Wordnet synset [8] are grouped together. The head noun is extracted as the super concept for this group of nominal n-grams. Wordnet hypernyms are used to guide organization of concepts into correct hierarchal relationships. Web-based part-of-speech error correction identifies and guides correction of some types of errors. A supervised clustering algorithm uses pseudo-relevance feedback as training data to create the concept hierarchy iteration by iteration.

Our research is tested on two corpora of public comments.

[1]http:\\regulations.gov

[2]The terms *ontology*, *taxonomy*, and *browsing hierarchy* are often used interchangeably. We use the term *ontology*.

Dg.o '08 Montreal, Canada

One corpus is comments about a proposal in 2007 by the U.S. Fish and Wildlife Service to list the polar bear as threatened ("the polar bear rule"). The other corpus is comments about a proposal in 2004 by the U.S. Environmental Protection Agency concerning mercury emissions from power plants ("the Mercury rule") [16]. Each corpus has about 500,000 email messages.

The remainder of this paper is organized as follows. Section 2 gives an overview of related research. Section 3 describes the procedure to get concept candidates for the ontology. Sections 4-7 introduce the procedure and techniques to build a concept ontology based on the concept candidates. Section 8 presents experimental results. Section 9 concludes.

2. RELATED WORK

Most concept ontology building efforts are manual [8] or semi-automatic [9, 11, 12, 14, 17, 19]. For the manual approaches, Wordnet [8] and the Open Directory Project [3] are two excellent examples. They are created and maintained by human experts. They are general knowledge about many kinds of entities and events. Because of abundance of human expertise, the accuracy of manually constructed concept hierarchies is usually high. However, subjectivity is unavoidable. Moreover, to re-create and modify the ontology is time-consuming and labor-intensive.

Semi-automatic approaches often use automatic techniques for the simpler subtasks, for instance, using part-of-speech (POS) tagging, verb predicate extraction [14, 19], or topic centroid extraction [9] to identify concept candidates. The rest of the work is done mainly manually. For example, manually selecting the concepts from a large pool of candidates, grouping the concepts by card sorting and laddering, and manually naming the clusters [14, 9, 19]. Sometimes, even for the simpler subtasks, there are also manual techniques to extract concept candidates based on term frequency and document frequency [11].

Automatic methods of creating concept ontologies [2, 3, 4, 5, 6, 7, 13, 15] also exist. To extract the concept relationships such as acronyms, hyponyms and hypernyms, Degeratu and Hatzivassiloglou [6] used surface text pattern matching to find them. Sanderson and Croft [15] found terms' hyponyms and hypernyms by studying the document frequency of terms. They considered a term x is term y's parent in the ontology hierarchy if the conditional probability of y given x $p(x|y) \geq 0.8$ and conditional probability of x given y $p(y|x) < 1$. Cimiano et al. applied Formal Concept Analysis [3, 4] to construct concept object and feature lattice, and identify common features shared by different objects as super-concepts for those objects. Hierarchical clustering is widely used in automatic methods of constructing ontologies. For instance, [13] and [6] used bottom-up hierarchical clustering, [3] used bi-section k-means divisive clustering, and [7] used a hierarchical self-organized map to group the concepts. Structure learning from Bayesian network is also used to to represent the ontology by Colace et al. [5].

3. CONCEPT CANDIDATES

[3]http://dmoz.org/

Mercury Dataset	Polar Bear Dataset
# terms in 2gm vocab(unique):462.	# terms in 2gm vocab(unique):1389.
# tokens in 2g m vocab: 4510.	# tokens in 2g m vocab : 3397.
power plants 370	greenhouse gas 248
mercury pollution 250	gas pollution 227
mercury emissions 175	sea ice 143
mercury levels 110	ice habitat 117
clean air 70	endangered species 115
# terms in 3gm vocab(unique):129.	# terms in 3gm vocab(unique):361.
# tokens in 3g m vocab: 900.	# tokens in 3g m vocab: 731.
clean air act 65	greenhouse gas pollution 227
environmental protection agency 35	sea ice habitat 115
air quality rule 30	endangered species act 104
pollution contr ol technology 25	arctic sea ice 62
interstate air quality 25	greenhouse gas emissions 19

Figure 1: Top bigrams and trigrams.

To create an ontology from a given text corpus, the first task is to find the candidate concepts. In large email corpora, there are many emails which are duplicates to each other since they are sent by email campaigns. By performing exact- and near-duplicate detection [20, 21], only unique comments are retained in the corpus and are ready for further analysis.

3.1 Nominal N-gram Mining

After duplicate detection, the non-duplicated parts of the corpus are obtained. Each passage is split into sentences by a sentence boundary detector. Each sentence is then parsed by a part-of-speech (POS) tagger from Stanford University.[4] The POS tagger annotates each word with its linguistic part-of-speech. For instance, the sentence "I strongly urge you to cut mercury emissions from power plants by 90 percent by 2008." is tagged as shown below.

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

An n-gram generator then scans through the parsed sentences to identify noun sequences, i.e., sequences of words tagged with NN(singular noun), NNP(proper noun), NNS(plural nouns), or NNPS(plural proper nouns). In the example above , "mercury/NN emissions/NNS" and "power/NN plants/NNS" are two such noun sequences.

Bigrams (2-word) and trigrams (3-word) are ranked by their frequency of occurrence. Figure 1 shows the top bigrams and trigrams generated from the Mercury and Polar Bear datasets.

Longer noun sequences, which can be considered as simple named entities, are also generated if they are a noun sequences with a capitalized first letter in every word. For example, sentence "The Toxics Release Inventory Burden Reduction Proposed Rule will deny information on the release of toxic agents." is tagged as:

The/DT Toxics/NNP Release/NNP Inventory/NNP Bur-

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

den/NNP Reduction/NNP Proposed/NNP Rule/NNP will/NN deny/VBP information/NN on/IN the/DT release/NN of/IN toxic/JJ agents/NNS ./.

In this example, "Toxic Release Inventory Burden Reduction Proposed Rule" is a named entity. These named entities and nominal n-grams are possible building blocks, i.e., concept candidates for the concept ontology construction.

3.2 Concept Filtering

Part-of-speech taggers make mistakes, especially when applied to text that has spelling, capitalization, punctuation and grammar errors, as is common in email. Tagging errors produce concept candidates that are not truly noun sequences. For example, the POS tagger may consider "protect polar bear" to be a sequence of three nouns, thus it becomes a candidate concept. POS tagging errors are a principal source of errors in the final concept hierarchy. More accurate tagging is desirable, of course, but tagging errors are inevitable with the current state of the art, thus a reliable way to detect and correct the errors is needed.

One common source of tagging errors is a verb or an adjective preceding a legitimate noun phrase. A second common source is unusual (often ungrammatical) word sequences, for example "cause cause". Word sequences misidentified as noun sequences typically have a lower frequency of occurrence than legitimate noun phrases. One might expect that they could be identifed by frequency-based heuristics.

In this work, each concept is formulated as a query and sent to the Google search engine. The top 10 snippets of the search results are extracted by regular expressions. Among the first 10 snippets, if a concept appears more than a threshold number of times (set to 4 in our case), it is considered to be a valid phrase and is kept, otherwise it is considered to be an error and is removed.

This heuristic is relatively effective at identifying and removing errors introduced by POS tagging mistakes. Some types of spelling errors, for example "polor bear" and "pulution", can also be removed by this method.

4. INITIAL CONCEPT HIERARCHY

The bigram concept candidates are organized into groups based on the first sense of the head noun in Wordnet. If the head nouns for two concepts are from the same Wordnet synset in their first sense (including the case that two concepts having the same head noun), then the two concepts are considered in the same group.

For these little groups, one of their common head nouns will be selected randomly to be added into the initial concept hierarchy as a parent concept. For example, "water pollution" and "air pollution" are two concepts which share a common head noun "pollution", therefore "pollution" is extracted to be their parent concept in the ontology.

The trigram or named entity concept candidates are compared with the bi-gram concepts already in the hierarchy. If a bigram concept matches with the suffix of a trigram or

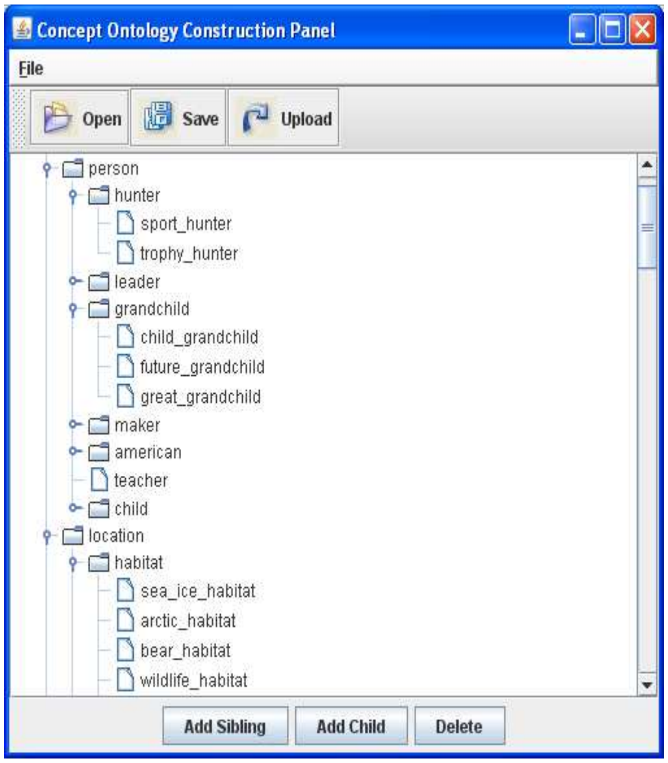


Figure 2: A Snapshot of Initial Concept Ontology

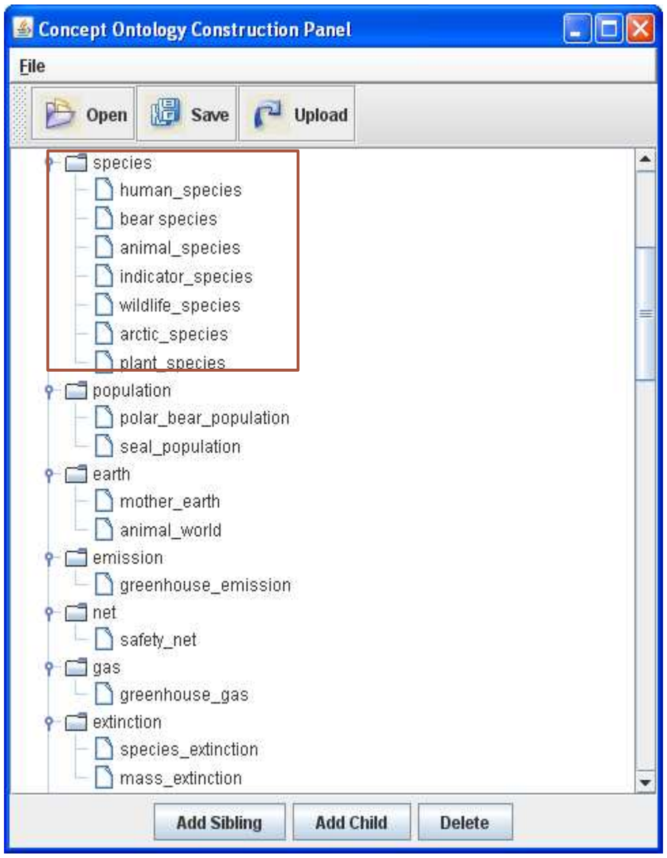
named entity, the trigram or named entity concept is added as a child of the bigram concept.

The above process is mainly based on head noun matching and substring matching. It creates a reasonable initial framework for the concept hierarchy. Figure **??** shows a snapshot of the application which starts the ontology construction process as detailed above. Figure **??** elaborates the ontology constructed for the "Polar Bear" dataset. The rest of the work is to enhance and refine the concept ontology based on it.

5. HYPERNYMS

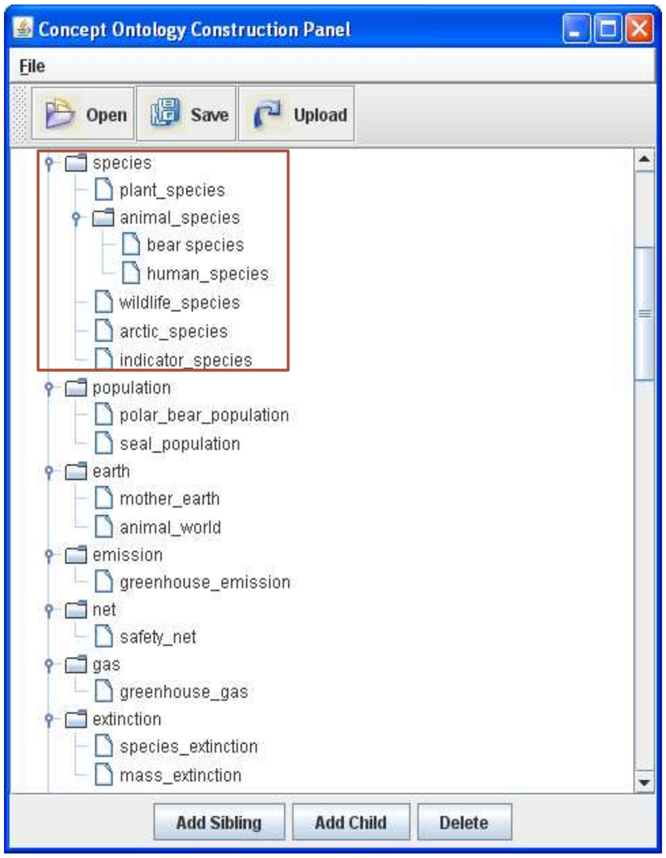
The string-matching based grouping creates little concept hierarchies (*hierarchy fragments*) such as, "species" with two children "animal species" and "bear species" as shown in Figure 3. In this case, however, the sibling, "animal species" is actually the hypernym of "bear species", thus it is in the wrong place. This type of error is common, so further refinement of each hierarchy fragment is necessary.

Concepts at the leaf level of each hierarchy fragment are looked-up in Wordnet. If one concept is another's hypernym, the former is promoted as the parent of the latter's. This process creates an intermediate level between the original parent concept and the leaf level, hence some hierarchy fragments are of depth two, and some are of depth three. In the aforementioned "species" example, we now have "species" with one child "animal species" and "animal species" with one child "bear species". It is shown in Figure 4.



- species
  - human_species
  - bear species
  - animal_species
  - indicator_species
  - wildlife_species
  - arctic_species
  - plant_species
- population
  - polar_bear_population
  - seal_population
- earth
  - mother_earth
  - animal_world
- emission
  - greenhouse_emission
- net
  - safety_net
- gas
  - greenhouse_gas
- extinction
  - species_extinction
  - mass_extinction

Figure 3: Another Example of Initial Concept Ontology



- species
  - plant_species
  - animal_species
    - bear species
    - human_species
  - wildlife_species
  - arctic_species
  - indicator_species
- population
  - polar_bear_population
  - seal_population
- earth
  - mother_earth
  - animal_world
- emission
  - greenhouse_emission
- net
  - safety_net
- gas
  - greenhouse_gas
- extinction
  - species_extinction
  - mass_extinction

Figure 4: A Snapshot with Hypernym Detection

At this point in the process some concepts are in different hierarchy fragments, but should have hyponym/hypernym relationship. Such concepts must be identified and connected together. A similar process as above is then applied to concepts in different little hierarchy fragments as well. Concepts in the same Wordnet hypernym chain are then connected and grouped into the same hierarchy fragment.

6. CLUSTERING TO CREATE HIGHER LEVEL HIERARCHY

By performing the above steps, the ontology consists many hierarchy fragments with depth 2-3, which makes the ontology a forest instead of a desired tree. Note that those hierarchy fragments are with high precision since they are produced by surface text patterns and Wordnet synonyms and hypernyms, which are usually able to generate high precision results. To further group the hierarchy fragments into higher level hierarchy, first, the distance scores need to be obtained among the roots of the hierarchy fragments. After that, a clustering algorithm can be used to group them into clusters. Finally, a naming algorithm should be used to assign meaningful names to the higher level concepts. The following sub-sections give details of the procedure.

6.1 Learning Similarity Scores

The hierarchy fragments existing in the ontology are with high precision. We assume that those concepts already in the hierarchies are correctly grouped. Therefore, the training instances are created by assigning 1 as the score to any two concepts within the same group in a hierarchy and 0 as the score to any two concepts which are not in the same group. Let concept c_i, c_j be any two concepts at level-n in the ontology. The similarity score e_{ij} for a training instance pair (c_i, c_j) is:

$$e_{ij} = 1, \text{ if } c_i \text{ and } c_j \text{ in the same cluster;} \\ 0, \text{ otherwise.}$$

By assigning scores to the training pairs in this way, we collect the training instances at the second highest level. The testing instances are the nodes at one level higher, that is to say, the nodes at the top level. The task here is to learn a similarity score Σ_{ij} between any two testing instances (t_i, t_j).

The estimated pairwise similarity score of both training and testing instances can be represented by a linear combination of some underlying features, such as similarity of Web definitions of two instances, similarity of sub-concepts, similarity of verb usage of two instances, etc. Let F_{ij} to be the vector containing the pairwise features for every two instances (c_i, c_j). The the pairwise estimated similarity score of two concepts can be represented as $\phi^T F_{ij}$. Our objective is to minimize the squared error of the training scores and the

```
Algorithm 1: K-medoids Clustering
1. Initialization. Randomly choose K instances as cluster
   centers.
2. Assignment. Assign each instance to the cluster which
   center is the most similar center to the instance.
3. Re-center. For each cluster, choose the instance
   maximizing the total similarity to other instances in the
   cluster as the new center.
4. Repeat step 2 and 3 until the assignments do not change.
```

```
Algorithm 2: K-medoids Clustering with Sampling
1. Sampling. Repeat K-medoids clustering in Algorithm 1
   for 1000 times.
2. Assignment. For each instance, assign it to the most
   frequent center related to it.
3. Re-center. Each center associated with non-empty cluster
   is set as a new center.
3. Sort. Sort clusters by size in descending order.
4. Finalize. For the instances belong to the first K clusters,
   retain the assignments. For other instances, assign them to
   the most similar one from the first K cluster centers.
```

estimated scores. Therefore, we have the following objective function:

$$\min_{\phi,E}(e_{ij} - \phi^T F_{ij})^2 + \lambda(E_{ij} - \phi^T F_{ij})^2 \quad (1)$$

$$E \succeq 0$$

where λ is a weighting coefficient and set to 0.5 in our system. $E \succeq 0$ is a constraint set on the estimated similarity score $\phi^T F_{ij}$ to make sure that it follows the regularity of a valid similarity score, i.e., to be positive semi-definite. The joint optimization can be done by any standard semi-definite programming (SDP) solvers. Our experiments make use of matlab software packages sedumi[5] and yalmip[6] to do the optimization.

By learning the coefficient vector ϕ, an estimated similarity score $\phi^T F_{ij}$ can be obtained for any given testing instance pair (t_i, t_j). The score also needs to be projected onto the SDP cone and therefore the final learned similarity score for (t_i, t_j) is Σ_{ij} to satisfy:

$$\min_{\Sigma}(\Sigma_{ij} - \phi^T F_{ij})^2 \quad (2)$$

$$\Sigma \succeq 0$$

The learned similarity score Σ_{ij} for testing instances (t_i, t_j) is used to group the instances into clusters.

6.2 K-medoids Clustering

Though ontology is a hierarchical structure, and eventually our system will produce a tree-like hierarchy as the ontology,

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

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

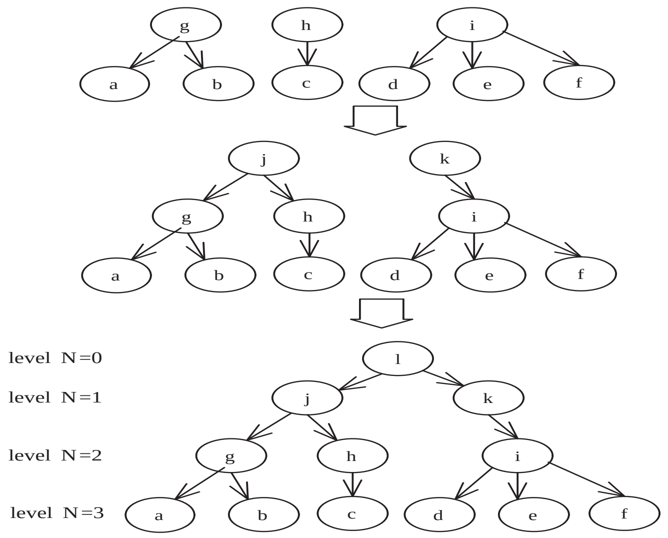


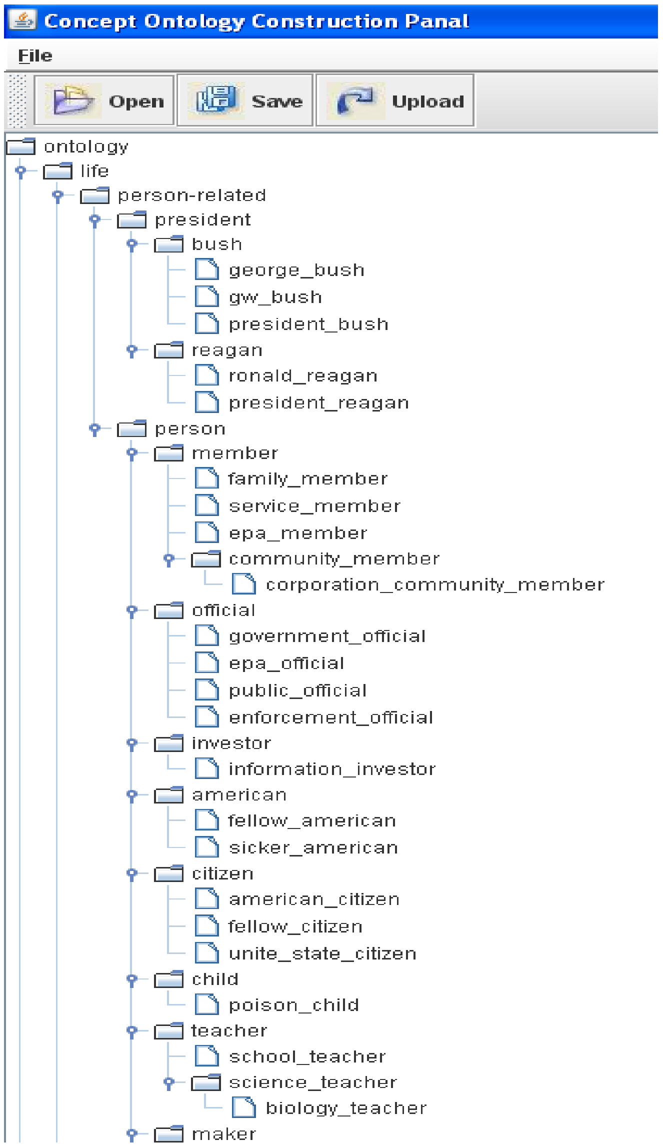
Figure 5: An Example Ontology

at each iteration, we can just do non-hierarchical clustering. That is to say, though we are doing hierarchical clustering, but as each clustering step, we are using a non-hierarchical clustering algorithm. The most effective non-hierarchical clustering algorithm is k-means. However, in our situation, it is not applicable since what we know is the pairwise similarity score instead of the actual score for each individual instance. Therefore, a more general version of K-means, K-medoids algorithm [10] is used here. Instead of using the calculated cluster centroid as in K-means, K-medoids use a real instance as the cluster center. Algorithm 1 shows the details.

The random initialization of the first K centers will lead to somewhat unstable assignments with different initialization. To get a more stable cluster assignment and to remove the randomness introduced by the random initialization in K-medoids algorithm, a sampling approach is taken. The above K-medoids clustering algorithm runs 1000 times for each K. Each time it is with a different random initialization. And then the most frequent assigned cluster center will be the new assignment for an instance. Note in this way, it is possible to produce a clustering assignment with more than K clusters, which is not desired. A slight modification to the sampling algorithm makes the correction. Algorithm 2 shows the details.

How to choose K, i.e., the number of clusters, is a traditional practical problem for K-means and K-medoids clustering algorithms. We adopt the Gap statistics [18] to estimate the optimal K. 20 sampled reference datasets are created for each distinct K. By comparing the expected changes of within-cluster-similarity between the null reference distribution and the actual distribution, in our experiment, the optimal K is around the values of one third of the total number of instances in each iteration.

6.3 Clustering with Pseudo-Relevance Feedback



```
ontology
  life
    person-related
      president
        bush
          george_bush
          gw_bush
          president_bush
        reagan
          ronald_reagan
          president_reagan
      person
        member
          family_member
          service_member
          epa_member
          community_member
            corporation_community_member
        official
          government_official
          epa_official
          public_official
          enforcement_official
        investor
          information_investor
        american
          fellow_american
          sicker_american
        citizen
          american_citizen
          fellow_citizen
          unite_state_citizen
        child
          poison_child
        teacher
          school_teacher
          science_teacher
            biology_teacher
        maker
```

Figure 6: Concept Naming

By K-medoids clustering with sampling, a hard cluster assignment is given for concepts at the same level in the ontology. Here the second highest level of those little high precision hierarchies are collected as training instances. For example, in Figure 5 the upper part, there are 3 hierarchy fragments. Nodes a-f form 3 clusters, (a,b), (c), and (d,e,f). According to section 6.1, we can obtain training instance pairs and their scores as $score(a, b) = 1$, $score(a, c) = 0$, $score(e, f) = 1$, etc. The testing pairs are (g,h),(g,i) and (h,i). By extracting the pairwise features and learning similarity scores for the testing pairs as in section 6.1, concepts g, h and i can be clustered into higher-level groups by K-medoids clustering algorithm described in section 6.2. For instance, in Figure 5 the middle part, concepts g, h, and i are grouped into (g,h) and (i). Note this process can be repeated many times until there is a single node of cluster found as the root. The entire process can be considered as a bottom-up procedure.

Table 1: Statistics of Polar Bear Dataset

Statistic	Count (before dup)	Count (after dup)
# documents	546,896	-
# sentences	-	224,573
# words	103,311,302	836,607
# unique terms	376,099	56,865

In each iteration, nodes at that second highest level in the current hierarchy can be extracted as the training instances, and to be used to learn the similarity scores for the highest level nodes. It is actually a pseudo-relevance feedback procedure with the assumption that the second highest level nodes are clustered correctly and hence can be used as the training data. Further extension to this work will be including human feedback in the loop and the system will be interactive and semi-automatic. So far, we just use pseudo-relevance feedback in the loop and maintain a fully automatic system.

7. CONCEPT NAMING

When two or more nodes combine to form a new group, there is a new node created as their parent. Unlike its children, this new node has no name. It is necessary to give the new parent node a good name representing the meaning of this entire group. A Web-based approach is taken in this work.

Again, we are going to use Google search engine as the provider of the knowledge to name the new concept. To cover the domain knowledge of all the child concepts, a query formed by concatenating the child concepts (with comma as the delimiter) is sent to Google search engine. The top 10 snippets returned by Google is parsed. After stopwords are removed from the snippets, the most frequent word is used as the name for the new parent.

For example, "Bush" and "Reagan" are grouped together, but there is no name for this group. By sending the query "Bush , Reagan", the system is able to find "president" as the name of this group of concepts. Figure 6 shows a snapshot of the ontology after doing Web-based concept naming.

8. EXPERIMENTAL RESULTS

To evaluate the resulting concept ontology, an evaluation scheme is designed as follows: For each pair of parent and child, extract them from the concept hierarchy, and compare them with a manually created concept ontology which is also in a form of parent and child pairs. The data sets used in the experiments are the entire "polar bear" comments (USDOI-FWS-2007-0008) for 9 weeks, which contains 546,896 email letters, and the entire "Mercury" comments (USEPA-OAR-2002-0056) , which contains 536,967 comments in total. Table 1 and 2 show the statistics of these two datasets before and after the duplicate detection.

8.1 Component-based Performance Analysis

Precision, recall, and F1-measure are three main metrics in the evaluation as in many Information Retrieval tasks. Precision is the number of correct <pa,ch> pairs divided by total number of such pairs returned by a ontology. Recall is the number of correct <pa,ch> pairs divided by total number of correct pairs in the human assessment. F1 is calculated as 2*Precision*Recall/(Precision+Recall).

Table 2: Statistics of Mercury Dataset

Statistic	Count (before dup)	Count (after dup)
# documents	536,967	-
# sentences	-	928,183
# words	104,564,253	6,403,742
# unique terms	574,784	64,289

Table 3: Component-based Precision, Recall and F1

Component(s)	Precision	Recall	F1
add 2gm	0.29	0.43	0.34
add 3gm	0.28	0.38	0.32
add WN hypernym	0.21	0.57	0.31
add POS corrector	0.35	0.58	0.44
add Supervised clustering	0.91	0.98	0.94
overall	0.91	0.98	0.94

From the descriptions in previous sections, we can see that that are several components in the system, namely nominal n-gram generator, hypernym detector, POS error corrector, and supervised clustering. We would like to know not only the overall system performance in terms of precision, recall and F1, but also the influence of each component to the system. This set of experiments are designed by adding the component one by one in the order of the concept ontology building procedure to see the add-on effect (Table 3) and by turning off a single component, keeping the rest to see the impact of each component on the overall system performance (Table 4).

From table 3, we observe that the POS corrector and supervised clustering greatly boost the performance of the system. POS corrector increases the precision by 14%, which is due to the fact that it removes POS errors, spelling errors and phrase recognition errors (for instance, ”bear bear” is considered as a noun phrase by the nominal n-gram generator). Supervised clustering improves the precision by 46%, which is a lot since the manually tagged ground truth uses the idea of super-concept as well, therefore many of the $< pa, ch >$ pairs are related to super concepts. By adding the supervised clustering, the system greatly improves in both precision and recall.

Table 4 shows the system performance by turning off the components one at a time to see which one has the biggest impact to the whole system. Figure 7 sorts the component by their impacts (measured in F1) in ascending order. The smaller the F1 value, the bigger the single component impact. Not surprisingly, bi-gram generator has the biggest impact, since it is actually the basis of the whole system to generate concept candidates. Supervised clustering also plays an important role since it introduces many super concepts. POS corrector is the third important component, it shows impact on removing wrong concepts. Wordnet hypernyms also helps to introduce more correct $< pa, ch >$ pairs. Tri-gram generator is the least important component here due to the fact that there are not that many tri-gram concepts appearing in the datasets.

Table 4: Component-based Precision, Recall and F1(by turning off each component)

Component(s)	Precision	Recall	F1
w/o 2gm	0.12	0.05	0.07
w/o 3gm	0.87	0.93	0.90
w/o WN hypernym	0.69	1	0.82
w/o POS corrector	0.57	0.97	0.72
w/o Supervised clustering	0.35	0.58	0.44
overall	0.91	0.98	0.94

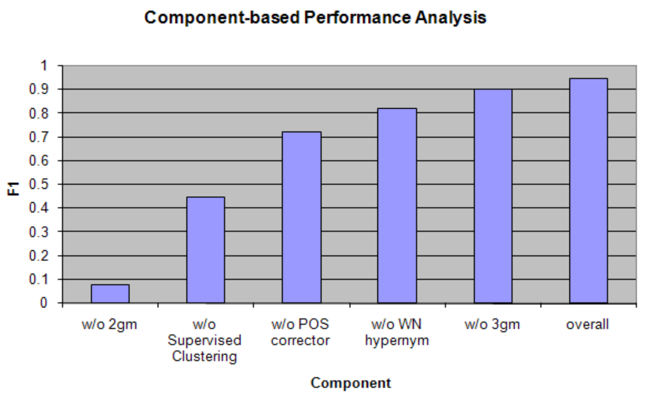


Figure 7: Component-based F1 values

Table 5: Component-based Error Analysis (by turning off each component)

Error Category	Overall	w/o POS Corr.	w/o Inter. Cpt
N-gram	5	4	1
POS	4	15	5
Address line	2	4	0
Noun Sequence	1	3	1
Polysemy	1	0	1
Supervised Clu.	1	1	24
Spelling	0	3	0
total	14/118	30/173	32/173

8.2 Error Analysis

Automatically building the concept ontology unavoidably produces errors. We perform an error analysis on ”polar

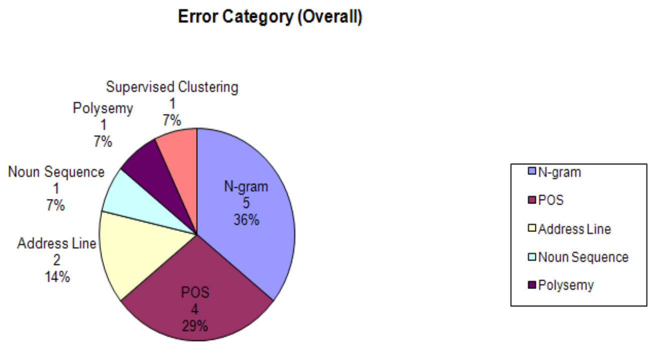


Figure 8: Error Categories

bear" dataset according to different types of errors in the resulting ontology, namely, n-gram generator error, POS error, address line parsing error, noun phrase recognition error, polysemy, inappropriate super concepts, and spelling error. Table 5 lists the number of errors in each error category for the overall system, the system without POS corrector and the system without supervised clustering. To study the error sources for the above three settings is because of that in the component-based experiment, we already know that POS corrector and supervised clustering are two of the most important components.

From table 5, we can see that POS corrector corrects 11 out of 15 POS errors, and 2 out of 4 address line parsing errors, 2 out of 3 noun phrase recognition errors and 3 out of 3 spelling errors. It is proved to be effective. Supervised clustering mainly helps with the super concepts - it introduces 23 new concepts. By doing all the error correction, there are still 14 out of total 118 concept pairs are errors. They are coming from n-gram generation (5), POS error (4), address line parsing error (2) and other errors. Figure 8 shows the percentage of each error category in the final concept ontology.

9. CONCLUSIONS

This paper details the techniques to automatically build a concept ontology for a given text corpus. By mining the nominal n-grams from the corpus, the candidates of the concepts are obtained. By incorporating Wordnet to identify hypernym relationships existing in the concept ontology, it creates more specific grouping of the concepts. Web-based part-of-speech error corrector removes the parsing errors in the earlier phrases, including address line parsing error, POS parsing error, spelling error and noun phrase generation error. Supervised clustering with pseudo-relevance feedback automates the entire process and creates the final concept hierarchy. We also gave component-based accuracy and error analysis to the system. The techniques used are shown to be effective in these experiments.

10. ACKNOWLEDGMENTS

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

11. REFERENCES

[1] J. Arguello and J. Callan. A bootstrapping approach for identifying stakeholders in public-comment corpora. In *Proceedings of the Seventh National Conference on Digital Government Research*. DG.O, May 2007.

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

[3] P. Cimiano, A. Hotho, and S. Staab. Comparing conceptual, divisive and agglomerative clustering for learning taxonomies from text. In *Proceedings of the European Conference on Artificial Intelligence*, pages 435–439. ECAI, 2004.

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

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

[6] M. Degeratu and V. Hatzivassiloglou. Building automatically a business registration ontology. In *Proceedings of the 2nd National Conference on Digital Government Research*. DG.O, 2002.

[7] D. Elliman and J. R. G. Pulido. Automatic derivation of on-line document ontologies. In *International Workshop on Mechanisms for Enterprise Integration: From Objects to Ontology, 15th European Conference on Object Oriented Programming*. MERIT, 2001.

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

[9] B. Fortuna, D. Mladenic, and M. Grobelnik. Semi-automatic construction of topic ontology. In *Conference on Data Mining and Data Warehouses*. SiKDD, 2005.

[10] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2001.

[11] A. Jaimes and J. R. Smith. Semi-automatic data-driven construction of multimedia ontologies. In *Proceedings of the 2003 International Conference on Multimedia and Expo*, pages 781–784. ICME, 2003.

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

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

[14] M. Sabou. Extracting ontologies from software documentation. In *Workshop on Ontology Learning and Population, European Conference on Artificial Intelligence*, pages 22–23. ECAI, 2004.

[15] M. Sanderson and B. Croft. Deriving concept hierarchies from text. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and Development in Information Retrieval*, pages 206–213. SIGIR, 1999.

[16] S. Shulman, J. Callan, E. Hovy, and S. Zavestoski. Language processing technology for electronic rulemaking: A project highlight. In *Proceedings of the Sixth National Conference on Digital Government Research*. DG.O, May 2005.

[17] C. J. Thomas, A. P. Sheth, and W. S. York. Modular ontology design using canonical building blocks in the biochemistry domain. In *International Conference on Formal Ontology in Information Systems*. FOIS, 2006.

[18] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a dataset via the gap statistic. In *Tech. Rep. 208, Dept. of Statistics, Stanford University.*, 2000.

[19] Y. Wang, J. Volker, and P. Haase. Towards semi-automatic ontology building supported by large-scale knowledge acquisition. In *In AAAI Fall Symposium On Semantic Web for Collaborative Knowledge Acquisition*, pages 70–77. AAAI, 2006.

[20] H. Yang and J. Callan. Near-duplicate detection by instance-level constrained clustering. In *Proceedings of the Twenty Ninth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, July 2006.

[21] H. Yang, J. Callan, and S. Shulman. Next steps in near-duplicate detection for erulemaking. In *Proceedings of the Sixth National Conference on Digital Government Research*. DG.O, May 2006.