An Atomatic Method for Constructing Domain-Specific Ontology Resources

Melania Degeratu, Vasileios Hatzivassiloglou

Department of Computer Science
Columbia University
1214 Amsterdam Avenue, New York, NY 10027, USA
{melania, vh}@cs.columbia.edu

Abstract

Data flow across multiple independent applications and further natural language analysis both require the establishment of a common foundation of terms and relations. Such a foundation can provide in-depth understanding of term equivalence within a domain sublanguage, and serve as a model of concept relations and dependencies. In this paper we discuss a domain-independent, corpus-based method for dictionary-less automatic extraction of ontological knowledge from domain-specific unannotated documents. We present the architecture, algorithms, and results for ONTOSTRUCT—a system that uses machine learning and statistical techniques to analyze text sources, discover terms, link equivalent terms into concepts, and learn both hierarchical and non-hierarchical conceptual relations. We report on ONTOSTRUCT's results in constructing domain-specific ontological resources and empirical evaluation of their quality.

1. Introduction

Ontology development has emerged as a complex research field involving knowledge representation, natural language processing, information extraction, knowledge acquisition, and text mining. Historically, ontologies have been manually generated, especially as it pertains to the discovery of equivalent terms and relations. The manual construction of large-scale ontologies such as WordNet (Miller et al., 1990) or CYC (Lenat, 1995) involves significant human effort and requires access to domain experts and specialist knowledge engineers. This process is hard to scale up, is not portable across different domains, and introduces the need for continuous ontology updating and maintenance. The overhead incurred by these operations can easily cause a manually constructed ontology to lose its consistency and currency. These problems and the emergence of readily available natural language processing tools have prompted researchers to develop semi-automated processes (Knight and Luk, 1994) and even explore fully automatic methods (Grefenstette, 1994; Sanderson and Croft, 1999; Bisson et al., 2000) for some ontology building subtasks. However, this automatization process relies on the use of existing ontologies, large corpora, online thesauri and dictionaries, and databases representing domain relationships. In this paper we present ONTOSTRUCT—a system that uses only the information encoded in unannotated natural language text to automatically perform all three component tasks for ontology construction: identify terms, link terms into concepts, and extract relationships between concepts. We explore methods for recovering terminological and semantic information from text sources without using any pre-existing knowledge resources such as dictionaries, term lists, database schemas, or semantic models of the domain.

2. Text Resources

The New Jersey State government has adopted an initiative in electronic commerce, with the goal of providing an effective and efficient electronic interface that fosters the establishment of new businesses and facilitates effective long-term interactive relationships with businesses throughout the State. An automatically constructed ontology can serve as both a shared language between distributed and non-coordinated software agents (the various programs collecting and validating registration data in each state agency) and an effective tool for constraining user input to acceptable values and regulating data flow.

We set out to develop automatic tools for the construction of such an ontology. The resources available to us are primarily texts that the agencies already use for supporting current business registration and regulation policies. These documents are not unlimited in number or especially large in size, so one direction that our research departs from other efforts is that we looked for techniques that would work with relatively sparse, but focused data. Unlike mining the web for generally reported information such as company-address relations (Agichtein and Gravano, 2000) or general semantic relationships of English words (Knight and Luk, 1994), we have a substantially more limited and non-extensible collection of documents. On the other hand, this textual data tend to have additional structure (e.g., lists of items, question and answer pairs, instructions for filling in blanks) that is not as prevalent in most text resources, and we have designed ONTOSTRUCT to take advantage of that partial structure.

For the experiments reported in this paper we have used pre-existing text documents provided by the state agencies, mainly business registration forms, permit application forms, and taxation forms, and their corresponding filling instructions. We have compiled a corpus of 101 PDF (Portable Document Format) documents, with a total of 181,748 words. Since the documents were developed by independent state agencies with no common structural and formatting guidelines, and they were intended for manual handling by both users and state employees, there is a great variety in their content, appearance, style, and structure, which adds on to the difficulty of the problem. As a consequence, we cannot develop a generalized method of extracting information based on the document's layout, and have to rely on weaker, local techniques for capturing structural information (e.g., by detecting itemized lists). Another important point arising from the analysis of this corpus is the

lack of multiple documents dealing with the same topic. While the documents are related, each of them focuses on information about a specific form, information that may or may not be generalizable to the other forms.

3. System Description

Constructing an ontology from natural language text entails term discovery, linking equivalent terms into concepts, and learning both hierarchical and non-hierarchical conceptual relations. The OntoStruct architecture contains modules that perform each of these steps.

3.1. Preprocessing

The input to our system consists of text documents in different formats. We provide modules that translate PDF or HTML documents to ASCII text. PDF document translation is achieved by automatically submitting the files to the online conversion tool available at the Adobe Acrobat Accessibility web page (`http://access.adobe.com/onlinetools.html`). The documents returned by this tool approximate the logical reading order of the text, and reformat it into plain text with significant loss of formatting information. We use a locally trained version of MXTERMINATOR (Reynar and Ratnaparkhi, 1997) to detect sentence boundaries, and align the text one sentence per line. We deal with the loss of formatting information by trying to recover list markers and captions for the fields present in the text. For this task we developed a bootstrapping algorithm that looks for increasing sequences of seed "counter" words (e.g., $1, 2, 3, \ldots$ or $a, b, c, \ldots$), and learns patterns that extract this type of sequences. These patterns serve as list markers and the process correctly identifies occurrences of list markers such as "section", "paragraph", "question", or "item" and marks them in the text. The extracted list markers provide valuable information towards topic detection, as they divide the text into topic-centered paragraphs. Since these paragraphs can also be nested, we extract a tree-like structure containing local structure information. We detect the lowest level in the nested list structure that two given terms share, and use this information as an additional indicator of the relatedness between the two terms.

Subsequently we use a part-of-speech tagger and chunker (Finch and Mikheev, 1997) to assign tags to content and function words, and to identify noun and verb phrases in the sentences. The tagger assigns to each word its locally most probable part-of-speech tag and the chunker uses these tags and a collection of finite-state automata to identify non-recursive noun and verb phrases. We also record the derivational root for all the verbs in the text. The purpose of these steps is to conflate all linguistic forms of a term in order to facilitate matching.

The final step in the preprocessing is to mark relative clauses and appositions, using finite-state grammars. The data collected so far is the input data for the subsequent modules of ONTOSTRUCT.

3.2. Term Identification

During this stage, the system extracts candidate terms from the pool of noun phrases identified in the corpus by the chunker used in the preprocessing stage. For each such noun phrase we use syntactical and morphological filters to compute a standardized form, which contains only the singular form of the nouns and their adjectival modifiers. Because ONTOSTRUCT aims to construct a domain-specific ontology for any given test domain, we choose not to eliminate adjectives from the standardized form of the noun phrases; for example, in the business registration domain our system learns that different provisions apply for "domestic profit" than for "foreign profit", and there are separate instructions for "professional corporations" and "nonprofit corporations". Therefore, we elect to accept as candidate terms all the variants we come upon in the text where the value of the *mutual information* (Fano, 1961) between the modifier and the head noun is positive.

We record the counts of all standardized candidate noun phrases that pass this cohesion test, and we select as terms all those noun phrases that appear at least twice in one of the documents in our collection, or appear in multiple documents. Such distributional characteristics have been used successfully before for term identification (Justeson and Katz, 1995).

3.3. Extracting Relationships

The ontological relations we want to extract are both hierarchical and non-hierarchical. In order to derive conceptual relations between the terms extracted from our corpus, we first analyze various syntactic patterns occurring in general text. Since we wish to apply our system to different text sources from diverse genres, the templates we considered are not developed specifically for one domain. Instead, ONTOSTRUCT uses general patterns—predefined or learned by bootstrapping in other domains, to retrieve *is-a* relationships, equivalences, general attributes of terms (including *has-a* or *part-of* relationships), and other general (unnamed) relationships between terms.

The extraction of hierarchical relations from text using predefined lexico-syntactic templates and patterns learned by bootstrapping has been established by (Hearst, 1992). Two examples of such patterns are *<TERM> is a <HEAD-TERM>* and *<HEAD-TERM> such as <TERM> [(, | and | or) <TERM>]**. Terms in lists, as in the last example, are likely to be specialized concepts of the noun phrase denoted *<HEAD-TERM>* in the pattern, representing instances of a hierarchical relation. One example of such a matched pattern is *New Business Entity* **(** *corporation, limited liability company, limited partnership* **or a** *limited liability partnership* **)**.

Using parenthetical patterns, we can also extract equivalence relations between two terms, or a term and its acronym. The acronym of a multi-word term is easily obtained in many cases by putting together the first letter from each term word. Examples of acronyms extracted with this method are "federal employer identification number (FEIN)" and "standard industrial code (SIC)". Other equivalence relations can be extracted using patterns like *<TERM-1> (also|formerly) called <TERM-2>*. Since a hierarchical relation is by definition asymmetric, all pairs of terms that appear in dual hierarchical relations are considered equivalent.

Term attributes and properties can also be extracted us-

ing contextual information (Berland and Charniak, 1999). We detect attributes that appear in the text in one of the forms *<ATTRIBUTE> of <TERM>* (e.g., "name of person") or *<TERM>('—('s)) <ATTRIBUTE>* (e.g., "person's name"). We find occurrences of these patterns from the text, and we create lists of attributes for all the terms in the corpus. Part of the list of attributes for the term *business* is {*address*, *assets*, *corporate name*, *offices*, *tax identification number*, *type*}.

The most frequent relations in free text are characterized by predicative links. Terms are predicatively linked to a verb to which they serve as subject or object. We extract relations matching the patterns *<TERM> <VERB>*, *<VERB> <TERM>*, and *<VERB> <PREPOSITION> <TERM>*. For each term in the corpus we then compile a list of verbs for which the term plays the role of the subject, and a list verbs for which the term is the direct object of the verb.

We use this contextual information to extract semantically related classes of terms (Grishman and Sterling, 1994), and extrapolate hierarchical information between these conceptual classes, as described in the next subsection.

3.4. Term Clustering

Different terms that are equivalent or nearly equivalent in meaning are often used by different data sources, in our case different state agencies or article authors. OntoStruct contains a module which detects term equivalences and forms conceptual classes on the basis of the similarity of the relationships in which extracted terms participate. For every term x, we start by compiling lists of the attributes assigned to that term and of the verbs where this term appears as a subject or object, producing three lists, *attribute_of*(x), *verb_with_subject*(x), and *verb_with_object*(x). We measure dissimilarity between each of the three pairs of corresponding lists L_x and L_y of two terms x and y with the Lance and Williams' (Lance and Williams, 1967) coefficient:

$$c(L_X, L_Y) = \frac{|L_x - L_y| + |L_y - L_x|}{|L_x| + |L_y|}$$

This calculation is carried out separately for the three pairs of lists (attributes, subjects, and objects) and the results are averaged to obtain the average *lexical* dissimilarity between the two terms. We multiply this lexical dissimilarity metric with a measure capturing *structural* information, specifically $1/d$, where d is the lowest level in our hierarchical representation of list structure in the documents where the two terms occurred together. If two terms have never been seen together in the same document or in the same component of a list structure, $d = 1$. If however the two terms co-occurred in a list item, their lexical dissimilarity is discounted proportionally to the specificity of that list item (we approximate specificity of list items by the level of nesting in the list).

Calculating the composite lexical and structural dissimilarity between all pairs of n terms results in a $n \times n$ dissimilarity matrix D. We apply the complete-link clustering algorithm (Frakes and Baeza-Yates, 1992) to this matrix to obtain a partition of the set of terms. Initially, every term is placed in its own cluster. We define the dissimilarity between clusters C_i and C_j as

$$d(C_i, C_j) = \max_{p \in C_i, q \in C_j} D(p, q)$$

(Obviously, initially the cluster dissimilarities are equal to the term dissimilarities as each cluster contains exactly one term). We then iteratively select a pair of clusters to merge. This is done in a greedy fashion, by picking the pair of clusters with minimum dissimilarity. These two clusters are merged, and the dissimilarities between the newly merged cluster and all other clusters are updated; this can be efficiently accomplished with the formula

$$d(C_k, C_l \cup C_m) = \max(d(C_k, C_l), d(C_k, C_m))$$

The complete link method is used to ensure the quality and tightness of the clusters; it is a good choice for clusters expected to have hyper-ellipsoid shape. Usually, a hierarchical clustering algorithm runs until all entries are encompassed by a single cluster. However, given the diversity and potential sparsity of the data, we chose to stop the process when the number of clusters is halved and limit the size of the clusters to a threshold value. Examples of clusters obtained using this clustering algorithm are {*manager*, *applicator*, *owner*, *manufacturer*, *worker*} and {*treasurer*, *president*, *secretary*, *comptroller*, *chief executive officer*, *corporate officer*, *designated agent*, *vice-president*, *vp*}.

3.5. Creating the Hierarchy

The classes of semantically related terms obtained as the result of the clustering algorithm and the hierarchical and equivalence relations learned in Section 3.3. are part of the input for the hierarchy construction algorithm. The output of the algorithm will be a forest of hierarchies that form an acyclic graph. The initial directed graph is obtained from the set of hierarchical *is-a* relations and is updated by merging all nodes with equivalent term labels. We traverse the graph in a depth-first fashion and we add to each node the similar words found by the clustering algorithm which do not appear on the same graph path.

The performance of this step depends on the quality of the output of the clustering module and on the number of hierarchical relations that match term clusters. Therefore we are generally able to generate only small trees, as in the example shown in Figure 1. The trees are then enriched by adding attributes to each node. Since hierarchical relations propagate attributes through inheritance down to the leaves, the number of attributes for the nodes in the resulting forest of trees will increase with the tree depth.

4. Results and Evaluation

For the business registration corpus OntoStruct extracted 12,607 terms out of 45,890 detected noun phrases. It found 314 hierarchical links, 1,575 attributes, and 208 equivalence relations. In addition, 667 terms were grouped into 222 conceptual classes. There were 23 non-trivial trees in the linked representation of conceptual classes. Note that in comparison to other text mining systems, the ratio of extracted information (terms and relations) to the number of

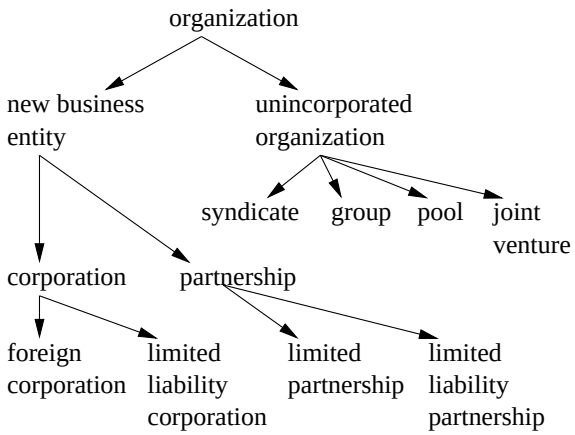

Figure 1: Sample hierarchy generated by ONTOSTRUCT.

words in the corpus is relatively high; this is a consequence of both the semi-structured information present in the forms and articles and our design decisions to extract information with relatively few instances of strong evidence.

We have measured the quality of OntoStruct's results for two of the most crucial modules (relationship finding and clustering). Currently, we evaluate only precision (as recall is very hard to estimate without a reference standard). For our test domain we select random test sets of 100 hierarchical, attributive, and equivalence relations produced by the system, and 100 term clusters. We presented human judges with domain information and definitions for the different kinds of relations whose correctness they had to assess, and we also gave them sample sentences where instances of each relation occur. Our results in Table 1 and Table 2 indicate that OntoStruct achieves precision of 71–83% in relationship extraction with Kappa agreement scores of 0.717–0.779, and obtains perfect clusters of terms 54% of the time. This makes a strong case for the algorithm's ability to learn regularities in the data and to discriminate against artificially introduced examples (due to errors in pattern matching and text preprocessing).

5. Conclusions

We have described OntoStruct, a fully automatic system for the learning of ontological information including terms, relationships, conceptual groupings of terms, and ontology tree fragments linking concepts. OntoStruct is successful in producing thousands of terms and relationships from a relatively small corpora of domain text, and our evaluation indicates that the extracted information is of high enough precision to be useful for automated ontology construction, perhaps with a further stage of output validation by a domain expert. The produced ontological resource can then be used in applications such as term alignment, input validation, and inference.

6. Acknowledgments

This work was supported in part by National Science Foundation awards IIS-9983468 and IIS-0306838. We are thankful to Nabil Adam and our collaborators at Rutgers University for facilitating access to the data and experts from the New Jersey State agencies.

	judge1	**judge2**	κ
is_a	70%	73%	0.779
attributive	81%	85%	0.717
equivalence	73%	68%	0.736

Table 1: Precision and agreement for relations.

	judge1	**judge2**
all correct	54%	49%
mostly correct	21%	19%
half correct	9%	1%
mostly wrong	3%	7%
all wrong	13%	15%

Table 2: Precision for term clusters.

7. References

Agichtein, E. and L. Gravano, 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the 5th ACM Digital Libraries Conference*.

Berland, M. and E. Charniak, 1999. Finding parts in very large corpora. In *Proceedings of the ACL-1999*.

Bisson, G., C. Nédellec, and D. Cañamero, 2000. Designing clustering methods for ontology building: the Mo'K workbench. In *Proceedings of the First Workshop on Ontology Learning at ECAI-2000*. Berlin.

Fano, R.M., 1961. *Transmision of Information: A Statistical Theory of Communications*. MIT Press.

Finch, S. and A. Mikheev, 1997. A workbench for finding structure in texts. In *Proceedings of the 5th ANLP*.

Frakes, W. and R. Baeza-Yates, 1992. *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall.

Grefenstette, G., 1994. *Explorations in automatic thesaurus discovery*. Kluwer Academic Publishers.

Grishman, R. and J. Sterling, 1994. Generalizing automatically generated selectional patterns. In *Proc. COLING-1994*.

Hearst, M. A., 1992. Automatic acquisition of hyponyms from large text corpora. In *Proc. COLING-1992*.

Justeson, J. S. and S. M. Katz, 1995. Technical terminology: Some linguistic properties and an algorithm for identification in text. *Nat. Lang. Eng.*, **1**(1):9–27.

Knight, K. and S. Luk, 1994. Building a large knowledge base for machine translation. In *Proc. of AAAI-1994*.

Lance, G. and W. Williams, 1967. A general theory of classification sorting strategies. *Computer*, **9**:373–80.

Lenat, D. B., 1995. CYC: A large-scale investment in knowledge infrastrature. *CACM*, **38**(11):32–38.

Miller, G., R. Beckwith, C. Fellbaum, D. Gross, and K. Miller, 1990. Introduction to WordNet: An on-line lexical database. *Int. J. Lexicography*, **3**(4):235–312.

Reynar, J. C. and A. Ratnaparkhi, 1997. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the 5th ANLP*.

Sanderson, M. and B. W. Croft, 1999. Deriving concept hierarchies from text. In *Proceedings of SIGIR-1999*.