Semantic Annotation of Biosystematics Literature Without Training Examples

Hong Cui
School of Information Resources and Library Science, University of Arizona, 1515 E. First Street, Tucson AZ 85719. E-mail: hongcui@email.arizona.edu

David Boufford
Harvard University Herbaria, Harvard University, 22 Divinity Avenue, Cambridge, MA 02138. E-mail: david_boufford@harvard.edu

Paul Selden
Paleontological Institute, University of Kansas, 1475 Jayhawk Boulevard, Lawrence, KS 66045. E-mail: selden@ku.edu

This article presents an unsupervised algorithm for semantic annotation of morphological descriptions of whole organisms. The algorithm is able to annotate plain text descriptions with high accuracy at the clause level by exploiting the corpus itself. In other words, the algorithm does not need lexicons, syntactic parsers, training examples, or annotation templates. The evaluation on two real-life description collections in botany and paleontology shows that the algorithm has the following desirable features: (a) reduces/eliminates manual labor required to compile dictionaries and prepare source documents; (b) improves annotation coverage: the algorithm annotates what appears in documents and is not limited by predefined and often incomplete templates; (c) learns clean and reusable concepts: the algorithm learns organ names and character states that can be used to construct reusable domain lexicons, as opposed to collection-dependent patterns whose applicability is often limited to a particular collection; (d) insensitive to collection size; and (e) runs in linear time with respect to the number of clauses to be annotated.

Introduction

The ever-increasing user expectation of digital libraries and digital repositories calls for semantic-based access to information at a finer granularity than the document level. In the biosystematics domain, whole organisms are constantly being studied and described, and descriptions are being revised over time. Ongoing digitization and integration projects such as the Biodiversity Heritage Library (BHL; Biodiversity Heritage Library, 2008) and the Encyclopedia of Life (EOL; http://www.eol.org) add to the urgency of the need for semantic-based access. As of June 30, 2009, BHL has OCRed over 14 million pages of legacy literature. These documents are waiting to be parsed so the knowledge locked in the pages will become more accessible. EOL, aiming to bring to one place information about every known species, is planning to parse text documents to produce identification keys, which represent a much more preferable, semantic-based access to biosystematics knowledge than keyword-based searches.

Biosystematics literature often contains names, classifications, and descriptions of whole organisms. Many biodiversity data in the deep Web are indexed by scientific names. Named entity recognition algorithms for biodiversity domains are being developed (Koning et al., 2005; Sautter et al., 2007). Although scientific names can often serve as an identifier for an organism, detailed morphological (also called diagnostic) descriptions, however, may be even more informative. These quite formalized sections describe the characters (e.g., *shape*) and states (i.e., values of characters, e.g., *oblong*) of taxa. For example, *leaf blades spatulate, oblong, or obovate to oval* describes the shape of leaves of a plant.

Large sets of concepts and relationships annotated and then extracted from the literature provide a better basis for computer-aided knowledge discovery (e.g., data mining) than text documents themselves. Domain-specific tools, such as identification keys for biodiversity domains, may be created

Received July 2, 2009; revised September 1, 2009; accepted September 14, 2009

 • Published online 9 December 2009 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/asi.21246

The original description of a plant consists of three clauses:
Clause 1: Stems 4–20 cm, scabrous, pubescent, hairs 2–4 times as long as broad.
Clause 2: Leaf blades narrowly linear to lanceolate, 1.5–5 cm × 2–10 mm.
Clause 3: Petals dusty pink.

The annotated description of the plant:
<description>
<stem relief="scabrous" pubescence="pubescent" length="4–20 cm">stems 4–20 cm, scabrous, pubescent, <hair proportionality="2–4 times as long as broad">hairs 2–4 times as long as broad.</hair></stem>
<leaf_blade plane_shape="narrowly linear to lanceolate" area="1.5–5 cm × 2–10 mm">leaf blades narrowly linear to lanceolate, 1.5–5 cm × 2–10 mm.</leaf_blade>
<petal coating= "dusty" coloration= "pink" >petals dusty pink.</petal>
</description>

The original description of a brachiopod consists of four clauses:
Clause 1: Pseudodeltidium, chilidium commonly absent;
Clause 2: concentric ornament well developed, regular;
Clause 3: spines at low angle, rare on dorsal valve;
Clause 4: marginal ridges present.

The annotated description of the brachiopod:
<description>
<pseudodeltidium_and_chilidium count="absent"> Pseudodeltidium, chilidium commonly absent; </pseudodeltidium_and_chilidium>
<ornament spatial_pattern="concentric; regular"> concentric ornament well developed, regular; </ornament>
<spine orientation= "low angle" location= "rare on dorsal valve">spines at low angle, rare on dorsal valve; </spine>
<marginal_ridge count="present"> marginal ridges present. </marginal_ridge>
</description>

FIG. 1. Examples of morphological descriptions showing clause and character levels of annotations.

in a semiautomated and more efficient manner with semantically annotated morphological descriptions. Identification keys comprise organism characters organized in a decision-tree structure that can be used to identify a specimen. Keys are highly intellectual but at the same time extremely laborious to create, which is because all characters of all organisms within the scope of the treatise must be collected and aligned. The laborious part of the work may be designated to computers if descriptions are semantically annotated. Semantic annotation also makes it possible to compare descriptions of organisms by their characters, as opposed to by keywords, which may be included in a description by chance. A tool with such capability would be especially helpful for entity retrieval, as defined by Text Retrieval Conference (TREC, n.d), for quality evaluation of manuscripts or self-published Web pages, and for tracking revisions of descriptions over time, to name a few applications. With short but otherwise typical morphological descriptions of a plant and a brachiopod, Figure 1 illustrates the levels of annotation needed for these applications. In the example, elements such as <stem>, <leaf_blade>, and <pseudodeltidium_and_chilidium> represent clause-level annotations, which name the organ a clause describes. Attributes such as relief, length, etc. are character-level annotations, which describe characters of the named organ. Element names are defined in the *Categorical Glossary for the Flora of North America Project* (Kiger & Porter, 2001) and a glossary published as part of the *Treatise on Invertebrate Paleontology* (Moore, Teichert, Robison, Kaesler, & Selden, 1952–2008). Attribute names are defined in Phenotype and Trait Ontology (PATO; Phenotype and Trait Ontology, 2006) and the *Categorical Glossary for the Flora of North America Project* (Kiger & Porter, 2001). Attribute values are taken directly from the original description.

There are a number of obstacles that need to be addressed to annotate biosystematics descriptions to a desired accuracy and granularity: (a) biosystematics includes a large number of finely divided branches, and each branch employs a different vocabulary. For example, terms used to describe an ant are different from those used for a brachiopod or a plant. Even terms used to describe one plant family may differ from those used for another. Figure 2 illustrates the term differences among 10 plant families in Flora of North America (FNA; Flora of North America, 2006). Large dots in the figure indicate family descriptions and small dots indicate a genus or species description from the preceeding family. The "valleys" between two family descriptions indicate

DOI: 10.1002/asi

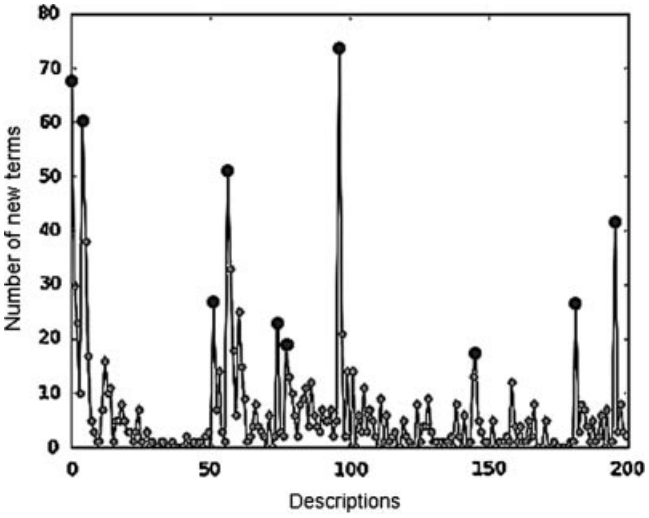

FIG. 2. Counts of new domain terms when descriptions of ten families and their inclusive genera/species from *FNA* are read sequentially.

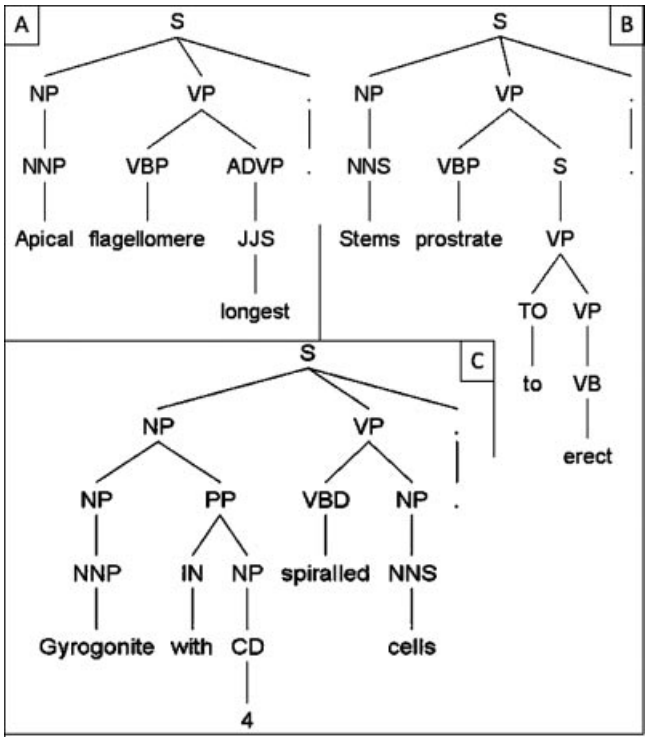

FIG. 3. Parsing examples of typical biosystematics clauses using the Stanford Parser.

that terms relevant to a family are used repeatedly in the descriptions for the genera and species within that family. The plot suggests that descriptions of the families contain different terms, because new term counts are high for family descriptions. (b) A good language processing infrastructure for biosystematics domains does not exist. A comprehensive, machine-readable dictionary or lexicon for a branch of biosystematics is often difficult to find, let alone a dictionary for the entire biosystematics domain. Although parsing tree banks such as PennTreeBank (TreeBank, n.d) and semantic networks such as WordNet (WordNet, n.d.) are effective language processing tools for general domains (e.g., Wall Street Journal articles), they do not provide good coverage for biosystematics domains and their counterparts for biodiversity domains do not exist. Figure 3 shows the parsing results of some typical sentences found in biosystematics descriptions using the Standford Parser, trained on PennTreeBank (StanfordParser, n.d). In Figure 3 panel A, "flagellomere," a noun, is mistaken for a verb (VBP), and "apical," an adjective, is mistaken for a noun (NNP); in Figure 3 panel B, the adjective phrase "prostrate to erect" is mistaken for a verb phrase; and in Figure 3 panel C, "4 spiralled cells," a noun phrase (NP), is split and parsed incorrectly. The results show that incorrect parts of speech are assigned and wrong attachments are made on those seemingly simple sentences. And (c) biosystematics descriptions are not written in standard English syntax, and there is not a standard syntax shared by authors from different branches of biosystematics.

This article presents an unsupervised machine learning approach that learns by exploiting the document content itself. The evaluation results suggest that domain lexicons and syntactic parsers may not be needed for clause-level annotation, as the document content itself holds the answers to most questions the learning algorithm needs to ask. Additionally, what the learning algorithm discovers from the documents may be used to construct domain lexicons or to enrich existing lexicons, such as WordNet, which can be very useful for other language processing applications of the domains.

This article is organized as follows. After a brief review of different approaches to semantic annotation, the unsupervised machine learning algorithm is introduced, where the emphasis is on how the algorithm exploits the content of a corpus to perform various subtasks of annotation. Performance evaluation of the algorithm on two real-life corpora from the domains of botany and paleontology are then reported, following which a discussion of the experimental results is provided, illustrating the strength and limitations of the algorithm. The article concludes with a discussion of future research directions.

Review of Relevant Literature

The research reported in this article is closely related to information extraction. Although they are both based on recognizing the semantic role a word plays, a typical task of information extraction is to fill out a template with key facts extracted from textual documents, while the goal of this research is to annotate complete descriptions of organisms. Since the mid-1980s, a great number of works in information extraction have been published, for example, Soderland (1999), Riloff and Jones (1999), Rokach, Romano, and Maimon (2008), and Rozenfeld and Feldman (2008), to name a few. While early works relied on hand-crafted rules, from the mid-1990s onward, machine learning became the

DOI: 10.1002/asi

dominant approach to learn extraction rules and to evaluate extracted results. As supervised machine learning needs training examples, which can be costly and time consuming to prepare, unsupervised (Riloff & Jones, 1999) or self-supervised methods (Rozenfeld & Feldman, 2008) have become more attractive under certain conditions. Such conditions include high data redundancy in extraction targets, relatively simple extraction targets (e.g., a company acquires another company), and the availability of large lexicons of the extraction domain, which provide semantically related terms and makes it possible to propagate the learning. For example, WordNet is a large-scale lexicon often used in supervised or unsupervised information extraction in general domains (e.g., Rozenfeld & Feldman, 2008). Probst et al. (2007) and Raju, Pingali, and Varma (2009) are the most similar to the work reported in this article in that they used semisupervised or unsupervised methods to extract product attributes from incomplete sentences without a predefined template. The differences include that they did not distinguish a part of a product (e.g., an LCD monitor in a cell phone) from a character of a product (e.g., the color[s] of a cell phone), and that they assumed that product attributes were explicitely mentioned in a description and when they were not, lookup lists were provided to the algorithm. Information extraction has also found its application in scientific domains, notably, biomedical domains. Although many issues in information extraction are common to all domains, different domains often face some special challenges. For example, recognizing negations in medical narrative reports is especially important to determine if a patient has a certain medical condition or not (Rokach, Romano, & Maimon, 2008). Text in the domain this article focuses on is marked by its heterogeneous terminologies, its deviated syntax from standard English, and its lack of related language processing utilities (e.g., lexicons). On the other hand, its syntax is much simpler than standard English syntax; for example, coreference resolution is seldom needed here. Below is a review of recent works related to biosystematics descriptions.

Research in Biosystematics Domains by Others

Lydon et al. (2003) shed light on the heterogenity of morphological descriptions. To develop a semantic markup system, Lydon and colleagues manually compared the descriptions of the same five species from six different floras and found that only 9% of the information across the source floras was exactly the same. Over 55% of the information was from a single flora, and around 1% of the information from different floras was contradictory. It is important to keep this variation in mind when designing a portable semantic annotation system that works across not only document collections of a domain but also domains of biodiversity, such as botany, zoology, and paleontology.

Research on semantic annotation of morphological descriptions has taken one of three approaches: the syntactic parsing approach, the rule-based approach, or the machine learning approach. Earlier projects (Taylor, 1995; Vanel, 2004) applied syntactic parsing methods to extract information from text to populate relational databases or to generate XML documents. While probabilistic parsing techniques have been around for the past decade, training examples (e.g., Penn TreeBank) and lexicons, which are necessary for successful parsing, were mainly developed for general domains, such as Wall Street Journal articles. As shown earlier, such parsers cannot be directly applied to biosystematics domains. Lacking these essential tools, the above-mentioned projects manually built small-scale parsers by examining the syntax and terms used in source plant descriptions. No scientific evaluation of system performance was reported by these studies. Although manually building syntactic parsers (including lexicon and grammar rules) on a collection by collection basis may attain high performance, it does not seem to be an efficient approach.

An example of a rule-based annotation system is GoldenGATE (Sautter et al., 2006), which is a specialized XML editor for biosystematics literature. GoldenGATE allows the user to invoke different built-in functions to paginate documents, tag taxonomic names, and tag paragraphs. The newest version supports annotation of citations. The end user, however, is free to come up with regular expression rules to annotate any aspects of the descriptions to any detailed level. While very flexible in accommodating documents from different collections/domains, GoldenGATE is a highly interactive system, requiring significant human intervention (e.g., correcting errors, tuning regular expression rules). The regular expression rules are sensitive to text variations and the need for the user to come up with such rules can limit GoldenGATE's application. In addition, how the rules crafted for one (set of) document(s) work on another one is unpredictable, hence the effort to fine tune the rules is largely wasted.

Similar to GoldenGATE, MultiFlora (Wood et al., 2004) adopted the General Architecture for Text Engineering (GATE; developed by the NLP research group at the University of Sheffield) for keyword searches and regular expression-pattern matching, but it also used an ontology, manually created by examining the selected descriptions from different floras (Lydon et al., 2003), and a hand-crafted gazetteer as a lookup list to link the extracted character states (e.g., "*oblong*") with one of the 134 identified characters (e.g., "*shape*"). While the work of Wood, Lydon, and colleagues shows that the descriptions from different sources are complementary, the authors acknowledge the "UnknownPlantParts" problem when reapplying the hand-crafted system to a new collection. The evaluation of the system on 18 species descriptions showed a recall and a precision in the range of mid 60 to mid 70. The research presented in this article may be useful to automate the manual knowledge engineering procedure to some degree so that the ontology-based approach can be better scaled up with larger data sets.

It has been shown that it is possible to learn regular expression patterns automatically using a supervised machine learning approach. Tang and Heidorn (2007) adapted Soderland's (1999) supervised learning algorithm to extract leaf shape,

DOI: 10.1002/asi

size, color, arrangement, and fruit shape characters from 1,600 of the *Flora of North America* species descriptions to fill out a predefined template, and they reported 30%–100% accuracy, depending on characters. Extracted information, even imperfect, was shown to increase user satisfaction with a full-text information retrieval system and improve user performance in specimen identification tasks.

Previous Research by the Authors

We (Cui et al., 2002) used automatic text classification algorithms, naïve Bayesian, and support vector machines, to annotate taxonomic descriptions at the paragraph level with satisfactory results (accuracy = 70%–100%). The goal was to identify types of paragraphs (nomenclature, description, discussion, distribution, references, etc.) in a taxonomic description. In another work, we (Cui and Heidorn, 2007) compared a homemade, association rule-based algorithm with naïve Bayesian and Support Vector Machine classifiers on a clause-level annotation in description paragraphs. Although the performance was good, both systems required an XML schema and 100s to over 1,000 annotated descriptions as training examples. Like Wood et al. (2004), our XML schema for clause-level markup was incomplete and some clauses had to be marked as "unknown features".

In summary, syntactic parsing and rule-based systems may be able to achieve high performance, but they need a significant manual knowledge engineering effort and can be collection or domain sensitive. Supervised machine learning has the advantage of better portability across collections and domains, but it requires a large number of training examples for each collection/domain, which could be difficult to obtain. The three approaches reviewed above have one thing in common: they take a top-down approach, in which some form of knowledge structure, whether lexicons, grammar rules, information extraction templates, or XML schemas, is predefined. The task of semantic annotation is essentially a task of fitting the textual descriptions in question into the predefined structure. Applying a top-down approach in a domain that is quite new to language processing could be problematic: lacking a well-defined, comprehensive, and machine-readable knowledge structure causes problems such as "UnknownPlantParts."

The approach presented in this article is a bottom-up approach, materialized by an unsupervised learning algorithm. Reported in the short paper of Cui (2008) is a preliminary version of the algorithm, which has since been expanded, further tested, and included as a key module (i.e., the *coreBootstrapping* module) in the complete algorithm reported here.

The Bootstrapping-Based Unsupervised Machine Learning Algorithm

The unsupervised machine learning algorithm looks mainly into the content of documents for guidance. The only external resource the algorithm accesses is WordNet. Since WordNet is readily available and has been shown to be useful for many language processing applications, our algorithm queries it occasionally when needed information cannot be obtained from the documents. The research question to be answered is, how much of the information needed for the algorithm to annotate morphological descriptions can be harvested from the descriptions themselves programmatically?

Bootstrapping is a type of unsupervised learning procedure that starts with a small set of "seeds" (i.e., known items, for example, organ names). The learning is achieved by using the seeds to iteratively discover new items (Riloff & Jones, 1999). Bootstrapping appears to be very effective with biosystematics literature, because the latter contains numerous concepts (organ names, characters, and modifiers) connected by relatively simple syntactic structures repeated under different contexts.

Doing away with training examples, the unsupervised algorithm employs various bootstrapping procedures to discover basic elements needed for semantic annotation from raw text documents. For clause-level annotation, which is reported in this article, the unsupervised algorithm learns organ names, character states, modifiers, and boundary words. These terms are defined below.

Definitions

The usage of some terms is clarified in this section. The descriptions of organisms comprise *clauses*, or sentences terminated by a colon, a semicolon, or a period. Most clauses do not have a verb, for example "basal leaves absent." In taxonomic documents, a typical clause comprises a subject (e.g., "basal leaves") and its descriptors (e.g., "absent"): The subject is typically a noun phrase, comprising a *modifier* or *"m" word* (e.g., "basal") and an *organ name* or *"n" word* (e.g., "leaves"), while the descriptor could start with any part of speech, such as adjectives or nouns. The word that starts a descriptor part is called a *boundary word* or *"b" word*. In the above example, "absent" is a "b" word. "length" in "resin gland length often much more than twice width" is another example of a "b" word. When the semantic role "n", "b" or "m" of a word is determined by the algorithm, the *word* is *tagged* with its role. Note that it is possible for a word to have multiple roles in different contexts. When words in a clause are sufficiently tagged, the algorithm is able to *annotate* the *clause* with a semantic tag (telling the aboutness of the clause), which includes an optional modifier and a head noun (typically an organ name). To avoid confusion, the tag that indicates the role a word plays is called *tag*, while the tag assigned to a clause is called *annotation*.

Boundary words are often values of characters (or "*character states*" in cladistic terminology), describing the features of an organ. For example, "elongated" in "leaves elongated" is a value (state) of the character "shape." Some boundary words are not character states, for example, "usually" in "roots usually flattened" is not a character state.

DOI: 10.1002/asi

While a modifier always appears before an organ name, a character state may appear before or after an organ name. For example "elongated" in "elongated leaves" is a character state and not a modifier, even though it appears before an organ "leaves." Sometimes the role of a word is context-dependent. For example "basal" in "basal leaves elongated" is a modifier because it narrows the scope of the subject to a subset of leaves (i.e., basal leaves), whereas "basal" in "leaves basal" is a boundary word and character state (i.e., position) because it states that all leaves are located at the base of the stem.

Note that these roles may be related to the part of speech of a word, but they are not parts of speech: a noun for example may be the name of an organ, but it may also be a modifier, for example "leaf" in "leaf blade" is a modifier.

Algorithm Overview

In clause-level annotation, the unsupervised algorithm determines the roles of the words in a clause until it gathers sufficient information to annotate a clause with a semantic tag. Examples of clause-level annotations are presented in Figure 1.

An overview of the algorithm is presented in Figure 4. Pseudocode of the modules marked with an * are presented in the Appendix. Modules marked with an "o" are optional depending on the characteristics of a source collection. Modules denoted with an "F" are used in annotating *Flora of North America* descriptions, while those with a "T" are used in annotating the *Treatise of Invertebrate Paleontology Part H* in the experiments reported in this article.

FIG. 4. Overview of the unsupervised learning algorithm.

The initiation step includes reading source descriptions from a directory, segmenting a description into clauses, and normalizing the text. A description is segmented by clause termination punctuation marks such as semicolon (;), colon(:), or period (.). As periods can be used for many other purposes, a sentence splitter module written by Yona (2002) was modified and used to avoid false segmentations. Specifically, periods following a single capital letter (e.g., as in a person's name J. Smith), in numbers (e.g., 7.8), and in abbreviated words (e.g., ca., var., diam., sq.) were ignored. The first two cases can be easily dealt with using regular expressions. As there were just a few examples of the third cases in morphological descriptions, the abbreviated words were hardcoded in the sentence splitter module, but they could be identified programmatically by finding all words that always appear before a period in a document collection. Segmented clauses were labeled with the name of their source files and with their location in the source file; for example, clause 111.txt-3 was the third clause in the file 111.txt.

Text normalization converts all uppercase letters to lowercase and standardizes the usage of hyphens. Unnecessary hyphens, for example, hyphens added to break a word at the end of a line, should be removed. Sometimes when raw text is extracted from a PDF or Word document, extra hyphens that are not in the original text may be introduced. Table 1 shows different ways a hyphen was embedded in the text after the text of *Flora of North America* volume 19 (*FNA* v19) was extracted from a Word document. To decide if a hyphen should be kept or removed, an algorithm was designed to connect as many pieces as possible to form a legitimate word and keep the hyphens connecting legitimate words intact. The algorithm checks against the corpus itself rather than a dictionary to decide if a string is a legitimate word. Because of the highly repetitive usage of domain terms in descriptions, the normal form of a broken word seems always locatable in the corpus. Details of this dehyphenization function are presented in the Appendix.

The pre-bootstrapping modules prepare the stage for the core bootstrapping modules to learn organ names, modifiers, and boundary words. Besides loading a number of finite sets of tokens such as stop words, prepositions, numbers, common prefixes (e.g., ob-, sub-, bi-), and suffixes (e.g., -ous) in biosystematics, there are three other important modules:

- *loadKnowledgeBase* module loads known words and their semantic roles if such knowledge is available. In the experiments reported here, no prior knowledge was loaded.
- *learnSeedNouns* module learns a set of plural and singular nouns from a document collection automatically, using some simple syntactic heuristics, one of which is: a word is a noun if its plural and singular forms are seen in the collection, but not its verb forms. For example, if "stem" and "stems" are seen in the collection but "stemming" or "stemmed" is not, then "stem" is believed to be a singular noun with a plural form "stems." Learned seed nouns are assumed to be organ

DOI: 10.1002/asi

TABLE 1. Usage of hyphens in extracted descriptions from *FNA* v19.

Usage	Hyphenated word(s) [standardized format]	Remark
Connect parts of a word	peti-oles [petioles]	Parts connected are not legitimate words.
	proxi-malmost [proximalmost]	Connect all parts to form a legitimate word.
	paniculi-form [paniculiform]	One part connected is a legitimate word. Connect all parts to form a longer legitimate word.
	pan-nose [pannose]	All parts connected are legitimate words. Connect all parts to form a longer legitimate word.
Connect multiple words	red-brown [red-brown]	Two words. Keep the hyphen.
	white-villous-ciliate [white-vilous-ciliate]	Three words. Keep the hyphens.
Hybrid usages	paniculi-form-scorpioid [paniculiform-scorpioid]	Different usages in one string. Remove the first but keep the second hyphen so that each word in the resultant string is legitimate and as long as possible.

names (i.e., "n" words), even though sometimes they are not. Details of this module are presented in the Appendix.

- *patternBasedAnnotation* module annotates clauses with distinct patterns, for example, in *FNA*, "x=" or "2n=" at the beginning of a clause is a sure indication that the clause is about chromosome counts of an organism. Some collections do not contain such patterns, for example, the *Treatise*. Different patterns are found in other collections, for example, in *Flora of China*, "Fl." stands for "flowering time." Descriptions of ants contain patterns such as "HL," standing for "head length." Patterns like these are most reliable cues for annotating some clauses; therefore, they should be applied first to move these clauses out of the way of the bootstrapping modules. The use of these patterns in formal publications such as *FNA* are typically described in Contributor's Guides. The FNA Guide was used to collect the patterns for FNAv19 for the experiments.

The core bootstrapping modules exploit a simple syntactic pattern seen in biosystematics documents: a simple clause starts with a pattern of "n b"—a subject followed by a boundary word.

- *coreBootstrapping* learns by making inferences between "n" words and "b" words. The following examples illustrate the learning process: given a clause "stems few," if "stems" is known to be an organ (i.e., "n," learned by the learnSeedNouns module), then "few" is a boundary ("b") word. Because now "few" is known to be a "b," given a clause "cauline leaves few," "cauline leaves" is then inferred to be the subject (i.e., "leaves" is the organ ["n"], and "cauline" is the modifier ["m"]). Similarly, the algorithm can in the future infer that "heads" and "branches" are organs when it finds clauses such as "heads few" and "branches few." Now that the module has learned that "stems," "leaves," "heads," and "branches" are "n," it can discover more "b" words by searching for clauses starting with these "n" words, for example, the clause "heads usually crowded" tells it that "usually" is a "b" word. Iterations like these go on until no new discoveries are made. At that point, this learning module terminates. Not all occurrences of a "b" or an "n" would warrant an inference. See the details of this module in the Appendix for the conditions under which an inference may be safely made. An early version of this module has been reported in Cui (2008).
- *leadWordBootstrapping* exploits the fact that the same organ of different organisms may have different descriptors (i.e., followed by different boundary words). For example, when seeing "stigmatic scar basal" and "stigmatic scar apical" and knowing "scar" is an "n," the algorithm may infer that "basal" and "apical" are two different "b" words. This module looks for such cues only at the beginning of a clause and helps to learn new words and roles that the *coreBootstrapping* module does not learn. Details of this module are presented in the Appendix.
- *unknownWordsBootstrapping* uses the fact that plural nouns are more often organ names ("n") than modifiers ("m") to infer that the word before a plural noun is a modifier, while the word after a plural noun is a boundary word. This module looks for such cues anywhere in clauses to tag the remaining unknown words. Details of this module are presented in the Appendix.

Learning about the roles different words play, the core modules use the subject as the tag to annotate each clause. The accurate role learning lays the groundwork for a set of secondary bootstrapping modules to deal with more complex language features in the biosystematics literature:

- *adjectiveSubjectsBootstrapping* deals with clauses that use an adjective for a noun as a subject, for example, "inner scarious" which should really be "inner [phyllaries] scarious" if its context were taken into account. In another context, "inner" may mean inner apex, inner floret, inner cypsela, etc. The module first identifies such usage of adjectives and then determines the organs they modify by tracing back to the last organ (i.e., parent organ) explicitly mentioned in the text. The module identifies such adjectives by exploiting the pattern in which a boundary word (e.g., "scarious") immediately follows a modifier (e.g., "inner") without an organ name in between. Details of this module are presented in the Appendix.
- *compoundSubjectsBootstrapping* deals with clauses that use conjunctions such as "and" and "or" to form a compound subject, for example, "stems, distal branches, and phyllaries gland-dotted." If stems, branches, and phyllaries are already tagged as an "n" and "gland-dotted" as a "b," then the module simply tags the clause as <stems and distal branches and phyllaries>. Otherwise, if the role of, say, branches is unknown, then the module would make a new inference that "branches" is an organ, just like "stems" and "phyllaries," because in English, "and" and "or" are used only to connect comparable tokens. Similarly, if all but the role of "gland-dotted" is

DOI: 10.1002/asi

known, the module can safely infer that "gland-dotted" is a boundary word.
- Because new discoveries may have been made by the above-mentioned modules, the *wrapupmarkup* module then uses the new knowledge to try to annotate the remaining unannotated clauses.

All bootstrapping modules carry out two functions simultaneously: one is to assign an appropriate role (organ name, modifier, or boundary) to an unknown word; the other is to annotate a clause if sufficient information is available. Whenever a noun is discovered, its singular and plural forms are added to the algorithm's knowledge base. Whenever a word, a noun or a boundary word, is learned, its prefixed or suffixed forms are also recorded. For example, when "acute" becomes known as a "b," "subacute" is tagged as a "b" as well; when "shrubs" becomes known as an "n," "subshrubs" is also tagged as an "n." A list of applicable prefixes and suffixes are loaded in the pre-bootstrapping stage.

Note that all bootstrapping modules run in iterations. New discoveries made at one point are immediately reflected in the algorithm's knowledge base and will affect the next decision point. Cases that cannot be handled by one iteration with confidence may become taggable in the next iteration when sufficient knowledge becomes available. Only reliable cues are used to make inferences, and the cues are applied in descending order of reliability, using the most reliable cues first to avoid introducing errors at early stages of iteration. This principle is applied throughout the algorithm. For example, *patternBasedAnnotation* is run before bootstrapping modules because those patterns are fixed and will not result in errors. The core bootstrapping modules are run before the secondary bootstrapping modules because the former deals with simpler clause structures, which contain less noise. Within *coreBootstrapping*, bootstrapping on clauses with plural nouns occurs before those with singular nouns, because plural nouns seldom play the role of modifier, while singular nouns may modify a head noun, or be head nouns themselves. Thus, chance of a wrong decision on plural nouns is much lower than on singular nouns. The order in which different patterns are checked in the *coreBootstrapping* module reflects the same principle. The implementation of this principle throughout the algorithm reduces errors and improves the accuracy of the annotation.

Upon completion of secondary bootstrapping, the machine should have learned enough about the words to annotate *directly* the remaining clauses. These clauses possess interesting language features that could confuse the algorithm if they were processed earlier:

- Clauses that are really phrases. For example, "biconvex, unisulcate valves;" Here, the head noun of the subject is the last word in the clause. These clauses are identified and annotated with the head noun by *phraseClauseAnnotation* module.
- Clauses that share a subject with an earlier clause. For example, "leaves cauline; sessile;" Here, the second clause "sessile's" subject is the subject of the first clause, "leaves." These clauses are identified and annotated with "ditto" by *dittoAnnotation* module.
- Clauses whose subject is either a pronoun or a character, as opposed to an organ name. For example, "maximum width at hinge line." These clauses are identified and annotated with "ditto" (as the subject of this clause is the last organ explicitly mentioned) by *pronounCharactersAnnotation* module.
- Clauses in which a comma is used to mean "and." For example, "adductor scars, brachial ridges weakly marked," which is semantically equivalent to "adductor scars and brachial ridges weakly marked," These clauses are identified and annotated with the compound subject (i.e., "adductor scars and brachial ridges") by *commaAsAndAnnotation*.
- Clauses in which character descriptors of an organ appear *before* the organ name. For example, "small cicatrix, rugae irregular, covering corpus," in which character state "small" describes the size of cicatrix and rugae but is used as if it were a modifier. The module *refineModifiers* attempts to distinguish these character states from true modifiers such as "ventral" in "ventral valve." If a word has only been tagged as "b," then the word is removed from a modifier. If a word has only been tagged as "m," then the word is kept as a modifier. When a word has been tagged as both "b" and "m," the module counts the number of times the word appears immediately before an "n" and the number of times the word appears after an "n." If the former is two times more than the later, then the word is kept as a modifier.
- The last module in post-bootstrapping is *normalizeAnnotations*, which converts all plural nouns in clause annotations (including an optional modifier and a head noun) into their singular form. This is done to avoid counting the same organ twice when determining the different organs described in a collection.

Post-bootstrapping modules act upon the role tags learned in the core and secondary bootstrapping to refine the annotations. Although there is no learning involved in these procedures, they are important in resolving cases that do not follow the simple patterns assumed by the bootstrapping modules. The bootstrapping modules assume a clause has an "m-n-b" pattern and try to recognize such patterns at the beginning of the clauses. Although this is the most frequent pattern in many morphological descriptions and provides reliable and recognizable cues, other patterns such as "b-m-n," "b, b-m-n-b," "b-b-b" do exist. Again, the algorithm learns from the simpler pattern first, and then applies what it has learned to other, more involved patterns. The "b-m-n" pattern is dealt with by *phraseClauseAnnotation*; "b, b-m-n-b" is dealt with by *commaAsAndAnnotation*; "b-b-b" is dealt with by *pronounCharactersAnnotation* and *dittoAnnotation*; and last, *refineModifiers* takes "b" words away from clause annotations. Taking bootstrapping and post-bootstrapping procedures as a whole, the simplified assumption on the syntax of descriptive clauses (i.e., "m-n-b" in lead words of a clause) is, in effect, relaxed.

Not all document collections will use all of the modules because of variations in community practices. For example, "adjectives as subjects" does not appear in the *Treatise* but does appear in *Flora of North America*, while "comma used as 'and"' does not appear in *Flora of North America*,

DOI: 10.1002/asi

TABLE 2. Basic statistics of *FNA* v19 and *Treatise* Part H.

	FNA v19 *Asteraceae*	*Treatise* Part H *Brachiopod*
Number of descriptions	942	2037
Number of clauses	12503	9755
Number of unique words	1957	2568
Number of organ names	243	492
Number of nonorgan name modifiers	53	68
Number of character states	1541	1353
Number of nondomain specific words[a]	≃20	≃655

[a]Number of nondomain specific words *approximately* equal to (total unique word – organ names – modifiers – character states) because some words could be modifiers and character states at the same time.

but appears in the *Treatise*. All modules can be turned on or off depending on the features of a collection.

Experiments

Data

The unsupervised annotation algorithm was evaluated on plant descriptions extracted from Volume 19 of *Flora of North America* (*FNA* v19 for short) and fossil brachiopod descriptions from Part H of *Treatise on Invertebrate Paleontology* (Moore et al., 1952–2008; *Treatise* Part H for short). *FNA* v19 contains 942 descriptions, while *Treatise* Part H contains 2,037 descriptions. To evaluate the robustness of the algorithm, it is important to test the algorithm against descriptions of different biosystematics subdomains. Table 2 shows basic statistics of the two volumes.

Method

1. Raw text was extracted from *FNA* v19 from a Word document. A homemade parser was used to extract descriptions from the text.
2. Descriptions of *Treatise* Part H were obtained from the author of a parser reported in Curry and Connor (2008).
3. The algorithm was executed on *FNA* v19 and *Treatise* Part H collection independently. The performance was recorded.
4. To study the effect of collection size on the performance, a learning curve was obtained for *FNA* v19 and *Treatise* Part H, respectively, by applying the following procedure:
   a. Randomly select descriptions from the data set in question to form subsets of descriptions of different sizes at an interval of about 500 clauses. By step 500, 24 subsets were randomly generated from *FNA* v19 and 18 were generated from *Treatise* Part H.
   b. Execute the algorithm on each subset and record the performance.
   c. Repeat step a and b three times and obtain the average performance on subsets of the same size.
   d. Use the average performances to plot the learning curve.

The algorithm was implemented in Perl and Java. The algorithm was run on a Gateway computer with Intel Core 2 Quad CPU, 2.40 GHz, and 1 GB RAM.

Performance was evaluated by comparing annotations made by the algorithm to those made by humans. Two benchmarks were prepared manually by the first author in consultation with the co-authors (who are a botanist/editor of *FNA* and a paleontologist/editor of the *Treatise* respectively) and other domain experts acknowledged at the end of the article. Clauses that may have different interpretations were presented to the domain experts for the correct or most likely interpretations. One benchmark was for *FNA* v19 and the other was for *Treatise* Part H.

The performance of the algorithm on the clause-level annotation was measured by accuracy. The accuracies of the learned modifiers, head nouns, and the annotations as a whole are reported separately.

$$\text{accuracy} = \text{clauses annotated correctly/total clauses}$$

The performance of the algorithm in learning word semantic roles was measured by precision and recall. Precision and recall of learned organ names were based on all organ names mentioned in the descriptions, no matter if they were the subject of a clause or not. The precision and recall of learned modifiers are reported for *Treatise* Part H as well, which evaluates the effect of *refineModifiers* module on distinguishing modifiers from character values. Because "boundary" is not really a semantic role, measuring its precision and recall would be meaningless. However, since boundary words should ideally be descriptors (i.e., characters or character states), measuring the percentage of learned boundary words that are really organ names is useful.

Even though character state learning was not the focus of the clause-level annotation, measurements were taken to evaluate the potential of the algorithm in learning character states, which are valuable for annotation at a finer granularity, i.e., character-level annotation. The measurements are (a) what portion of the learned boundary words are character states (i.e., precision) and (b) what portion of all character states are learned (i.e., recall).

To evaluate the effect of WordNet on performance, the following measurements were also taken: (a) the number of words posted to WordNet for a part of speech, (b) the number of times WordNet returned any result, and (c) the number of times WordNet returned multiple parts of speech for a word.

Results

Table 3 shows the algorithm's performance on the whole sets, of *FNA* v19 and *Treatise* Part H descriptions.

The experimental results on subsets of *FNA* v19 and *Treatise* Part H are reported side by side. Starting from subset one, which contains around 500 clauses, each subsequent subset has around 500 more clauses than the previous one. This set of results answers the question about the optimal size of a corpus for the algorithm to achieve optimal performance.

Figure 5 shows the sizes of the subsets, Figure 6 shows annotation accuracies of the algorithm on subsets of various

DOI: 10.1002/asi

TABLE 3. Performance of the algorithm on the whole sets of *FNA* v19 and *Treatise* Part H.

	FNA v19	*Treatise* Part H
Accuracy on modifiers	0.9773	0.9420
Accuracy on head nouns	0.9897	0.9705
Accuracy on annotation as a whole	0.9708	0.9232
Precision on organ names	0.9187	0.8876
Recall on organ names	0.9300	0.7866
Percentage of boundary words that are organs	0.0000	0.0027
Percentage of boundary words that are states	0.9766	0.9235
Recall on states	0.7573	0.4996
Number of total learned words	1722	1819
Number of WordNet requests made	215	529
Number of result received	37 or (37/1722 = 2.15%)	147 or (147/1819 = 8.08%)
Number of "multiple parts of speech"	17	75
Run time in minutes	45.27	71.63

sizes, Figure 7 shows the compositions of learned boundary words from the subsets, Figure 8 shows the WordNet access data, and Figure 9 shows the times taken for the algorithm to process the subsets.

Discussions

Discussion of Annotation Performance on the Whole Sets of FNA v19 and Treatise Part H

The experimental results (Table 3) show that the algorithm's performance on clause-level annotation is approaching human performance on *FNA* v19, with the accuracy of annotation as a whole exceeding 97%. On *Treatise* Part H, the clause-level annotation is quite accurate, with an accuracy on head nouns exceeding 97% and overall accuracy of 92%. The poorer performance on *Treatise* Part H is largely due to the algorithm's inability to distinguish some of the character states from modifiers, for example, "flaring," which appears in "flaring teeth," "flaring socket plates," "flaring dental plates," "flaring socket ridges," etc. Although a character state (a shape), because the term consistently appears before an organ name throughout the corpus, as opposed to being part of the descriptors following an organ name (e.g., "teeth flaring"), the algorithm mistook it for a modifier. Lower modifier accuracy causes lower overall annotation accuracy.

When extending the search scope beyond the subject of clauses, the algorithm's performance on learning organ names drops somewhat, with a 92% precision and 93% recall on the *FNA* v19 collection and an 89% precision and 79% recall on the *Treatise* Part H collection. The false organ names learned from *FNA* v19 and *Treatise* Part H are shown in

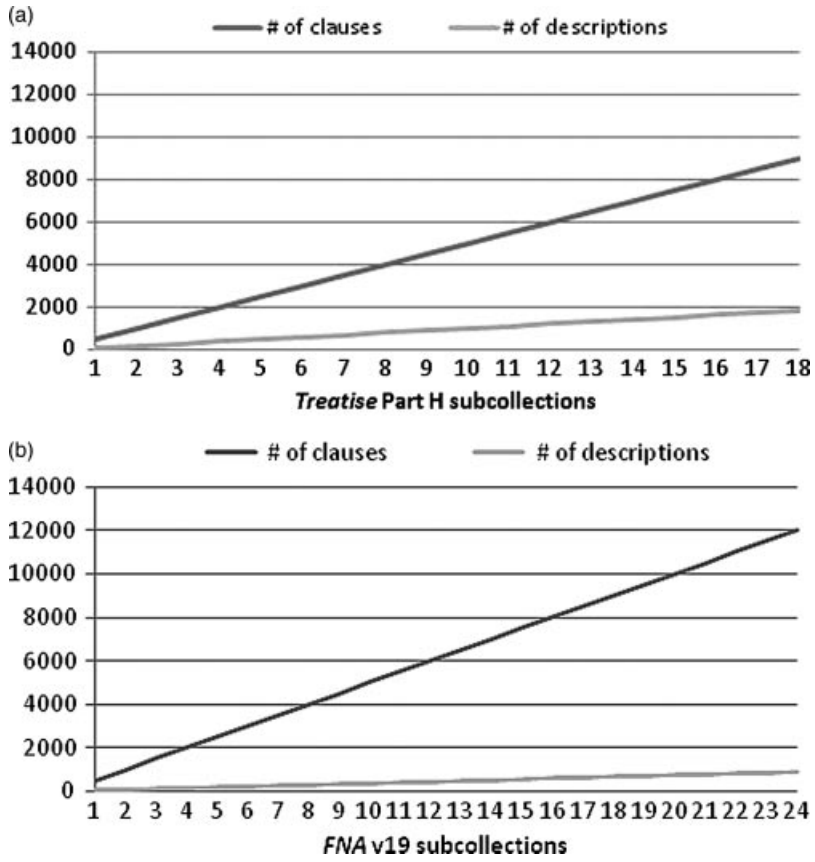

FIG. 5. Sizes of the subcollections of *FNA* v19 and *Treatise* Part H.

DOI: 10.1002/asi

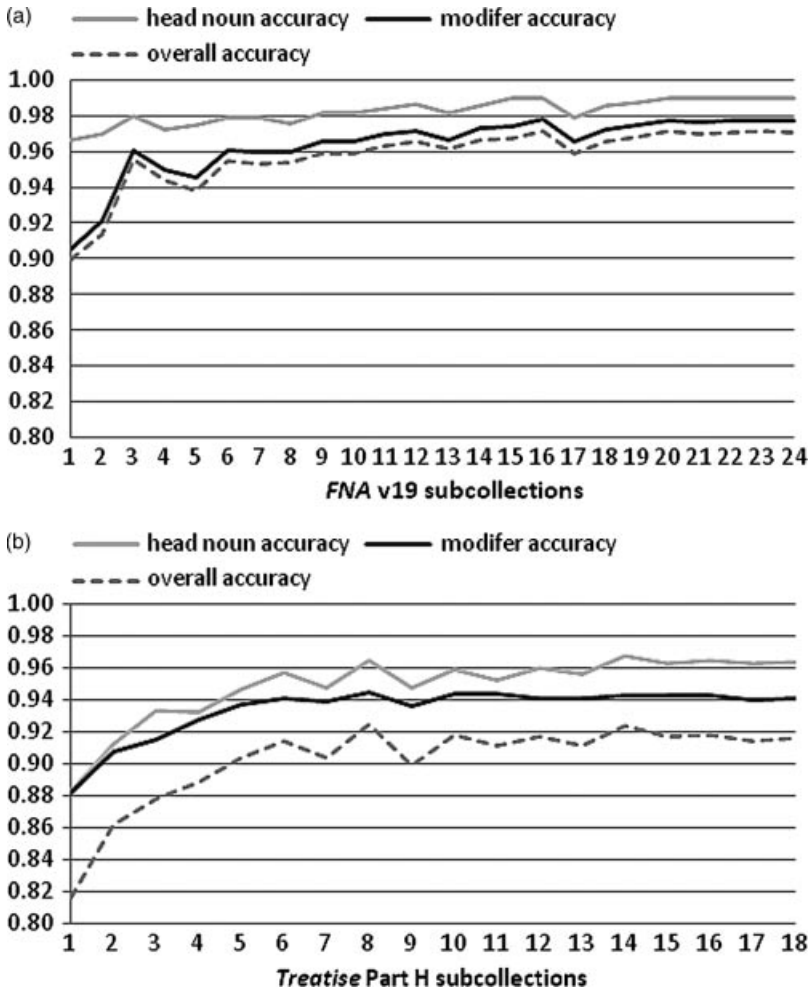

FIG. 6. Annotation accuracies of the algorithm on the subcollections of *FNA* v19 and *Treatise* Part H.

Table 4. Most errors were introduced directly or indirectly by the *learnSeedNouns* module, which assumes the nouns it learned are all organ names. For *FNA* v19, although words such as "combination" and "combinations" are nouns, they are not organ names. "bilateral" and "lateral" were tagged as organ names because "laterals" was seen in a description as a subject, so the algorithm inferred that the singular form "lateral" is also an organ, and that "bi-lateral" should have the same role as "lateral" because the prefix "bi-" should not alter the role the word plays. Because those false organs rarely appeared in or near the subject of a clause, they affected the annotation performance only slightly and were not detected by the *coreBootstrapping* module. If there were a clause like "stylopodia lateral" in the collection, these two errors may have been corrected by the *coreBootstrapping* module. When the module sees a phrase that forms a "ps" pattern ("stylopodia" is plural and "lateral" is singular), the module would change the role of "lateral" from a singular organ name (s) to a boundary word (b). *learnSeedNouns* also explains many of the errors seen in *Treatise* Part H. Words such as "angle," "angles," "degree," and "degrees" were correctly identified as nouns, but they were not organ names. There were also more verbs in *Treatise* Part H than in *FNA* v19. Most verbs were correctly treated as boundary words, but a few, for example "breaks," "dominates," "reaches," and "starts," were mistaken for plural nouns. In the *Treatise* descriptions, more non-domain specific words were seen in clauses, such as "characters as for family/subfamily" or "poorly known and some key characteristics not recorded," which, strictly speaking, are not morphological descriptions. All of these contributed to the lower performance on *Treatise* Part H.

Note that all the nouns mistaken for organ names are common English words, as opposed to domain terms. This suggests a possible fix to the problem: Use a general corpus such as Brown Corpus (Francis & Kucera, 1979) as a filter to identify and remove nouns that are not subjects of clauses. The non-subject condition may be necessary because words such as "leaves" should be considered both as a common English word and a domain term.

Boundary words learned from *FNA* v19 are almost exclusively words representing character states, accounting for 97.66% of boundary words. By identifying the boundary words alone, the unsupervised algorithm uncovers 75.73% of all words representing character states mentioned in the

DOI: 10.1002/asi

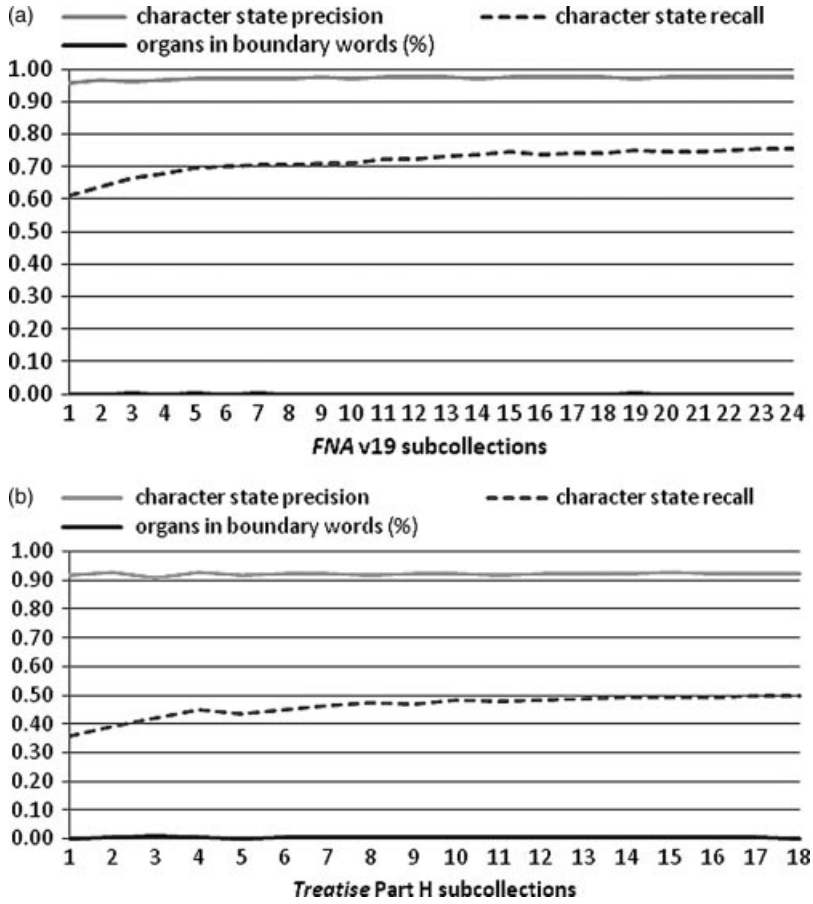

FIG. 7. Compositions of learned boundary words from the subcollections of *FNA* v19 and *Treatise* Part H.

description collection. For *Treatise* Part H, 92.35% of the boundary words are words representing character states, and the unsupervised algorithm uncovers 49.96% of all words representing character states mentioned in the description collection. Boundary words that are not related to character states are shown in Table 5. Note that adverbs such as "much" and "rarely" are considered to represent a character state because they are often used to modify the degree of a state, as in, for example, "much-branched." The overlap between boundary words and character states suggests that the boundary words may be used directly in character-level annotation, in which each character state should be tagged and associated with its character, its organ and taxon. This may be a feasible approach because few organ names were mistaken for boundary words.

The unsupervised algorithm determined the roles for 1,722 of the total 1,957 words (or 88%) in *FNA* v19. The algorithm needed to consult WordNet (v 2.1) on 215 (or $215/1722 = 12\%$) words, but $(215 - 37)/215 = 83\%$ of those requests generated no return because the word was not in WordNet. Of the 17% (or 37) successful requests, almost half ($17/37 = 46\%$) returned multiple parts of speech that needed to be resolved by the algorithm. On *Treatise* Part H, the algorithm determined the roles for 1,819 of the total 2,568 words (or 71%). The algorithm needed to consult WordNet on 529 (or $529/1819 = 29\%$) words, $(529 - 147)/529 = 72\%$ of those requests failed. Of the 28% (or 147) successful requests, more than half ($75/147 = 51\%$) returned multiple parts of speech that needed to be resolved by the algorithm. The higher success rate may be due to the existence of more common terms in *Treatise* (Table 2). The higher success rate did not seem to improve the annotation performance. This set of results suggests that WordNet's coverage of biological terms and WordNet's contribution to the algorithm's performance is limited.

Discussion of the Learning Curves Constructed From Subsets of FNA v19 and Treatise Part H

On the smallest subsets from either source, the unsupervised algorithm achieved over 80% accuracy on overall annotation (*FNA* v19: 90%, *Treatise* Part H: 81.2%). The annotation accuracies are higher if the modifier and head nouns are considered separately (Figure 6). When collection size increases, the annotation accuracies show an increasing trend. These results seem to suggest that the learning of the algorithm is rather effective and robust. It seems to be robust because the increased noise introduced by larger collection sizes did not drag down the annotation accuracy. The optimal performance is reached when collection size reaches 3,000 clauses.

DOI: 10.1002/asi

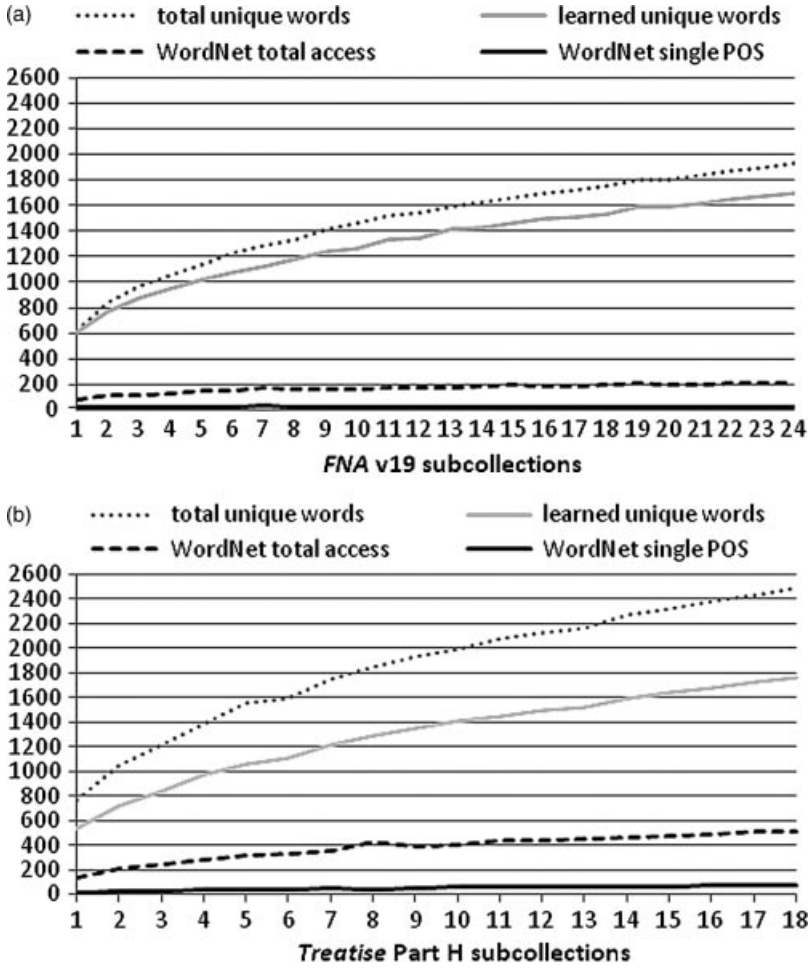

FIG. 8. Access of the algorithm to WordNet while annotating the subcollections of *FNA* v19 and *Treatise* Part H.

Starting from the smallest subsets, Figure 7 shows that the boundary words learned by the algorithm are mainly words related to character states. Few organ names are mixed with boundary words. The recall on character states increases with the size of the collections. The results seem to be consistent between the *FNA* v19 and *Treatise* Part H subsets. The high precision in character states learned as boundary words reinforces our initial impression that the learned boundary words may be valuable for annotation at the character level.

With collection size increasing, the number of unique words in a collection naturally increases, and so does the number of unique words the algorithm needs to learn in the process of annotating the clauses. The number of total words and unique words increases at roughly the same rate. In contrast, the rate at which the number of WordNet accesses increases is much flatter, and the number of a single part of speech returns remains almost constant. The results shown in Figure 8 suggest that by exploring the text alone, the unsupervised algorithm could gather a majority of the information it needed to perform the clause-level annotation. The data also confirms the earlier finding that WordNet's coverage of words in biosystematics is limited.

Last, Figure 9 suggests that the algorithm runs in linear time with respect to the number of clauses processed (*FNA*: $R^2 = 0.9898$ and *Treatise*: $R^2 = 0.9689$). This means the algorithm is not only effective, but also efficient in processing the two collections.

Overall, the algorithm performs rather well, given the fact that the algorithm knows nothing about either domain when it starts. The algorithm learns from the plain text descriptions a majority of the information needed to perform the clause-level annotation. The algorithm achieved higher precision than recall in a number of areas. High precision may be more desirable than high recall when both cannot be achieved at the same time. High precision ensures the learned modifiers, organ names, and character states are reusable in another bootstrapping procedure without introducing many errors, or for enriching a domain lexicon. Harvesting good domain terms from various sources as a way to improve recall is a preferable strategy (Wood et al., 2004), especially when the algorithm is inexpensive to run.

Conclusion

An algorithm is reported in this article for processing descriptive documents whose syntax differs from standard English syntax in its condensed use of nouns, adjectives, and adverbs for domain concepts and in its lack of verbs.

DOI: 10.1002/asi

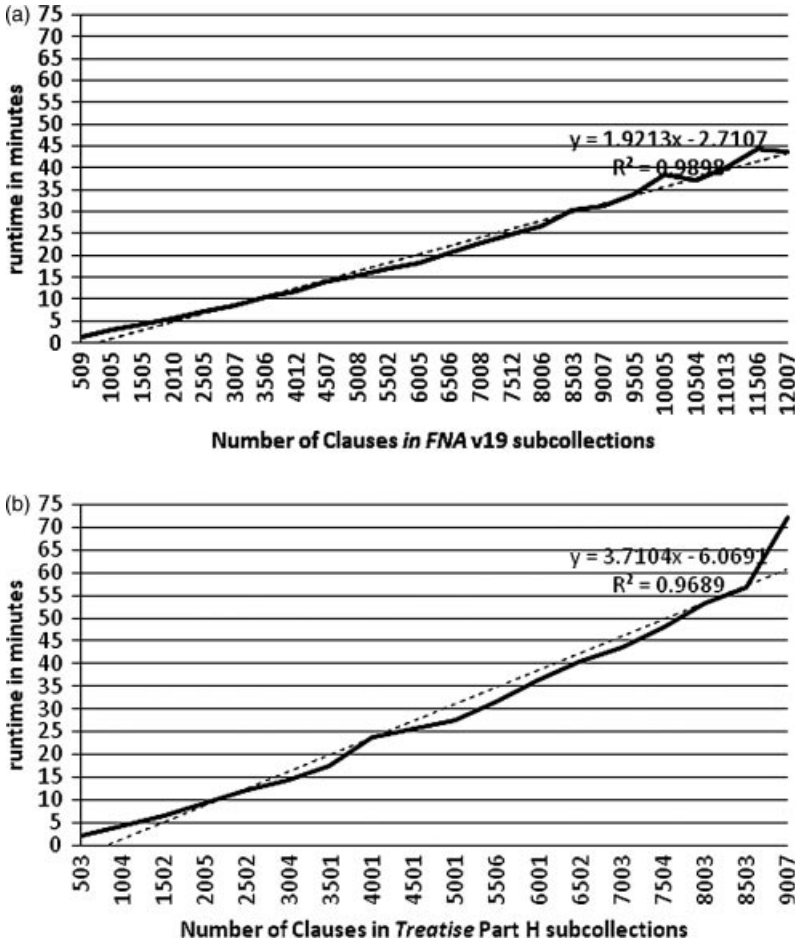

FIG. 9. Times taken for the algorithm to process the subcollections of *FNA* v19 and *Treatise* Part H.

TABLE 4. The complete list of words mistaken by the algorithm for organ names on *FNA* v 19 and *Treatise* Part H.

FNA v19	*Treatise* Part H
bilateral, center, color, combination, combinations, diam, diams, directions, divisions, dots, lateral, other, others, overtop, plane, ring, time, times, types, unit	adults, affinities, angle, angles, break, breaks, characteristic, characteristics, degree, degrees, description, descriptions, dimensions, disposition, dispositions, dominant, dominate, dominates, dubium, enclosure, families, family, growth, insertion, means, members, other, others, position, reaches, relationship, relationships, representatives, row, space, specimen, specimens, stage, stages, start, starts, structure, subfamily, top

TABLE 5. Some boundary words extracted from *FNA* v19 and *Treatise* Part H that are not related to character states.

FNA v19	*Treatise* Part H
again, age, ca, early, ever, except, excluding, gla, included, including, just, kinds, late, later, least, like, mainly, may, mound, never, northern, otherwise, perhaps, particularly, per, produce, produced, producing, range, reaching, remaining, replaced, resemble, resembling, seen, think, time, times, together, unlike, winter	accommodate, accommodating, acting, additional, aforesaid, alongside, elsewhere, also, although, apparently, appearing, aspect, assign, assigned, associated, assumed, available, away, because, belongs, certainly, characterized, chiefly, commonly, corresponding, creating, definitely, depending, described, determining, diagnosis

Such syntax is often seen in organism morphological descriptions. Because language processing utilities have not been developed for such documents, the algorithm takes a bottom-up approach to learn what the corpora themselves may offer.

The evidence from the set of experiments reported in this article suggests the following:

1. Biosystematics descriptions in deviated syntax hold ample cues that can be exploited by an unsupervised learning

DOI: 10.1002/asi

algorithm to distinguish the semantic roles of the words for tasks such as semantic annotation of the descriptive clauses. This article shows that the task of dehyphenizing, seed nouns learning, organ names and head nouns learning, and boundary words learning can all be performed rather effectively by looking for cues in the corpus content. The cues held by the plain text seem to be sufficient for the algorithm to perform with satisfactory or better accuracy. No training examples, domain lexicons, or grammar rules are needed. The removal of these prerequisites may considerably lower the barrier to the adoption of the technique. The domain concepts learned seem to be clean enough to be used to enrich or construct domain lexicons. Syntactic patterns or parsing trees of the clauses may be generated to tailor a syntax parser for the domain tasks.

2. The learning strategy works well, despite the syntactic variations between *FNA* v19 and *Treatise* Part H. The learning starts with the clauses that match "[organ name]-[boundary words]" pattern, which seems to give reliable cues—knowing a boundary word can reliably infer the organ name and vice versa under certain conditions. Not all clauses match this simple pattern, for example, a good proportion of clauses from *FNA* v19 used *adjectives* for a subject (e.g., "inner," "outer"), many clauses in *Treatise* Part H have an organ name *following* a number of boundary words (which are really character states), and both have many clauses *without* a subject. These variations are handled after the simple clauses have been processed and a good number of words accurately learned. The highly repetitive usage of domain terms (such as organ names and character states) works to the advantage of the algorithm, as the words learned from simple clauses appear again in other clauses, making it possible for the algorithm to annotate the latter with good accuracy as well.
3. Anticipating differences in syntax used in different corpora of descriptions, the design of the algorithm should remain modular. In the experiments reported here, certain secondary bootstrapping and post-bootstrapping modules were selectively used, as noted in Figure 4.
4. Although the setting was manual at this time, a way to automatically detect certain syntactic features of a corpus and active necessary modules is desirable and will be investigated in the future.

Ongoing and Future Work

Further development of the algorithm is underway to annotate *FNA* descriptions at the character level, which extracts character states from plain text descriptions and puts them in a format such as XML (as shown in Figure 1), the EQ format (Mabee, et al., 2007, for example, "cauline leaves [E: entity]/ovate [Q: quality]" , "quality" means character state), or a RDF triple (for example, "cauline leaves hasShape ovate"). Since characters may not be directly extractable from raw text because they are often implied rather than stated (e.g., in "cauline leaves ovate," the character "shape" is implied), ontologies like PATO (PATO, 2006) or domain experts must be consulted. Although existing ontologies may not cover all concepts discovered from domain text, what they already have can be used to prime the algorithm. In return, the algorithm can help enrich ontologies by discovering and proposing new concepts and properties. Character-level annotation could be complicated, however, by the fact that existing ontologies many not agree on the meaning of a character state. For example, the term "erect" takes on a number of different meanings depending on which ontology one consults: The FNA Glossary (Kiger & Porter, 2001) defines "erect" as a state of *orientation*, the Oxford Plant Characteristics (OPC, n.d) defines it as a state of *habit*, and PATO ontology it as a state of *position* (with a synonym *placement*). Without a shared view of the domain, the interoperability among different annotated collections is lost, defeating the purpose of semantic annotation.

Another piece of the puzzle is a module that can extract sections of descriptions from parent documents, whether a book or a journal article. Other methods have been reported, including our own (Cui et al., 2002), but with the unsupervised learning algorithm and the sets of organ names and character states learned from the botany domain, an easier method may be possible. This method involves constructing a "model" description containing learned organ and state names and using it to identify plant descriptions by simply checking to what extent words in a candidate text overlap with those in the "model" description. This strategy has been successfully applied to extract the description section from *FNA* v19 for this study. The same strategy may be applied to documents of other taxonomic groups.

Developing and evaluating the algorithm on a character-level annotation on a number of different corpora contributed by various biodiversity communities is planned. These collections include various plant families and fossil or living invertebrate animals (ants, wasps, etc.). The annotated collections will be used to populate biodiversity digital libraries, generate identification keys, and for data mining applications. With this system, we hope to mobilize the biosystematics knowledge locked in the text to support research in systematics biology, comparative biology, and ecology.

Acknowledgment

We thank the publishers of *FNA* and *Treatise* for permission to use the data sets in this project. We also thank Professor Richard Connor of the Department of Computer and Information Sciences of the University of Strathclyde, Glasgow, for giving us the source descriptions extracted from *Treatise* Part H, and Jill Hardesty of the Paleontological Institute of the University of Kansas and other anonymous domain experts for their contributions in making the benchmark for *Treatise* Part H. Last, we thank the anonymous reviewers for their constructive suggestions on revising the article.

References

Biodiversity Heritage Library. (2008). Biodiversity Heritage Library. Retrieved July 10, 2008, from Biodiversity Heritage Library: http://www.bhl.si.edu/

DOI: 10.1002/asi

Cui, H. (2008). Unsupervised semantic markup of literature for biodiversity digital libraries. Proceedings of the Eighth ACM/IEEE-CS Joint Conference on Digital libraries, (pp. 25–28). New York: ACM Press.

Cui, H., & Heidorn, P. (2007). The reusability of induced knowledge for automatic semantic markup of taxonomic descriptions. Journal of the American Society for Information Science and Technology, 58(1), 133–149.

Cui, H., Heidorn, P., & Zhang, H. (2002). An approach to automatic classification for information retrieval. Proceedings of the Second ACM/IEEE Joint Conference on Digital Libraries, (pp. 96–97). New York: ACM Press.

Curry, G., & Connor, R. (2008). Automated extraction of data from text using an XML parser: An earth science example using fossil descriptions. Geosphere, 4(1), 159–169.

Flora of North America Editorial Committee. (2006). Flora of North America North of Mexico guide for contributors. Retrieved July 10, 2007, from http://www.fna.org/FNA/Guide/guide_2006.pdf

Kiger, R.W., & Porter, D.M. (2001). Categorical glossary for the flora of North America project. Retrieved January 3, 2009, from http://huntbot.andrew.cmu.edu/HIBD/Departments/DB-INTRO/IntroFNA.shtml

Koning, D., Sarkar, I., & Moritz, T. (2005). TaxonGrad: Extracting taxonomic names from text. Biodiversity Informatics, 2, 79–82.

Lydon, S., Wood, M., Huxley, R., & Sutton, D. (2003). Data patterns in multiple botanical descriptions: Implications for automatic processing of legacy data. Systematics and Biodiversity, 1(2), 151–157.

Mabee, P., Ashburner, M., Cronk, Q., Gkoutos, G., Haendal, M., Segerdall, E., et al. (2007). Phenotype ontologies: The bridge between genomics and evoluation. Trends in Ecology and Evoluation, 22(7), 345–350.

Moore, R.C., Teichert, C., Robison, R.A., Kaesler, R.L., & Selden, P.A. (1952–2008). Treatise on invertebrate paleontology. Lawrence, KS: University of Kansas/Boulder, CO: Geological Society of America.

OPC – Plant characterisitics. (n.d.). Retrieved May 21, 2009, from http://herbaria.plants.ox.ac.uk/vfh/image/?glossary=show

PATO. (2006). Phenotypic quality ontology main page. Retrieved June 10, 2009, from http://www.bioontology.org/wiki/index.php/PATO:Main_Page

Probst, K., Ghani, R., Krema, M., Fano, A.E., & Liu, Y. (2007). Semi-supervised learning of attribute-value pairs from product descriptions. In M.M. Veloso (Ed.), Proceedings of the 20th International Joint Conferences on Artificial Intelligence (pp. 2839–2843). Rochester Hills, MI: IJCAI.

Raju, S., Pingali, P., & Varma, V. (2009). An unsupervised approach to product attribute extraction. In M. Boughanem, C. Berrut, J. Mothe, & C. Soule-Dupuy (Eds.), Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval. Lecture Notes in Computer Science, 5478, 796–800.

Riloff, E., & Jones, R. (1999). Learning dictionaries for information extraction by multi-level bootstrapping. Proceedings of the Sixteenth National Conference on Artificial Intelligence (pp. 474–479). Menlo Park, CA: Association for the Advancement of Artificial Intelligence.

Sautter, G., Agosti, D., & Bohm, K. (2006). A combining approach to find all taxon names (FAT). Biodiversity Informatics, 3, 46–58.

Sautter, G., Agosti, D., & Bohm, K. (2007). Semi-automated XML markup of biosystematics legacy literature with the GoldenGATE Editor. In R.B Altman, A.K. Dunker, L. Hunter, T. Murray, & T.E. Klein. (Eds.), Proceedings of the Pacific Symposium on Biocomputing (PSB 2007) (pp. 391–402). Hackensack, NJ: World Scientific.

Soderland, S. (1999). Learning information extraction rules for semi-structured and free text. Machine Learning, 34(1–3), 233–272.

StanfordParser. (n.d.). The Stanford Parser: A statistical parser. Retrieved June 30, 2009, from http://nlp.stanford.edu/software/lex-parser.shtml

Tang, X., & Heidorn, P.B. (2007). Using automatically extracted information in species page retrieval. Retrieved July 10, 2008, from http://www.tdwg.org/proceedings/article/view/195/0

Taylor, A. (1995). Extracting knowledge from biological descriptions. Proceedings of 2nd International Conference on Building and Sharing Very Large-Scale Knowledge Bases (pp. 114–119).

Text Retrieval Conference. (n.d.). TREC entity retrieval. Retrieved June 30, 2009, from http://ilps.science.uva.nl/trec-entity/

Treebank, P. (n.d.). The Penn Treebank Project. Retrieved June 30, 2009, from http://www.cis.upenn.edu/∼treebank/

Vanel, J.-M. (2004). Worldwide botanical knowledge base. Retrieved July 5, 2007, from http://wwbota.free.fr/

Wood, M., Lydon, S., Tablan, V., Maynard, D., & Cunningham, H. (2004). Populating a database from parallel texts using ontology-based information extraction. In F. Meziane, & E. M'tais (Eds.), Proceedings of Natural Language Processing and Information Systems, 9th International Conference on Applications of Natural Languages to Information Systems. Lecture Notes in Computer Science, 3136, 254–264.

WordNet. (n.d.). WordNet: A lexical database for the English language. Retrieved June 30, 2009, from http://wordnet.princeton.edu/

Yona, S. (2002). Sentence. Retrieved September 4, 2009, from http://cpansearch.perl.org/src/SHLOMOY/Lingua-EN-Sentence-0.25/lib/Lingua/EN/Sentence.pm

DOI: 10.1002/asi

Appendix

Details of Selected Modules

In this appendix, details of a number of selected modules are presented in pseudocode. These modules illustrate the ways the algorithm exploits the content of the documents itself to obtain the information needed to perform a certain task. In the pseudocode below, "$" is used to indicate a scalar variable, "@" to indicate an array, "@@" to indicate a two-dimensional array (i.e. a matrix), and "/* */" to include a comment.

Dehyphenation

Function: remove unnecessary hyphens from text using. For example, turn "paniculi-form-scorpioid" to "paniculiform-scorpioid."
Exploits: the occurrences of the word in its standard form in the text.
Input: a collection of plain text descriptions.
Output: updated text in which unnecessary hyphens are removed.

```
1. @hyphenatedwords = getherAllWordsFromCorpus
   Containing("-")
2. foreach $hyphenatedword in @hyphenatedwords{ /*e.g.
   $hyphenatedword = "paniculi-form-scorpioid"*/
3. @dehyphenatedword = (); /*create an empty array to
   save parts of the dehyphenated term*/
4. @parts = segmentByHyphen($hyphenatedword)
   /*@parts now holds ("paniculi," "form," and
   "scorpioid")*/
5. $size = numberOfElementsIn(@parts) /*$size now
   equals 3*/
6. @@matrix = initiateASquareMatrix($size) /* initialize
   a 3x3 square matrix; each column and each row of this
   matrix represent a part */ /*populate the upper-right half
   of the matrix as follows:*/
7. if (connecting parts from $row up to $column forms a
   legitimate word){ /* $row and $column takes the values
   from 0 to $size-1; $row <= $column*/
8.     $matrix[$row][$column] = 1
9. }else{
10.    $matrix[$row][$column] = 0
11. }
12. sort the rows in @@matrix in descending order of the
    distance of 1 to the diagonal line
13. foreach row in @@matrix{
14.    $dehyphenatedword = connect the parts whose cor-
    responding value is 1
15.    add($dehyphenatedword, @dehyphenatedword) /*add
    $ to @*/
16.    remove the rows representing connected parts from
    further consideration
17.    exit the loop when no part is left
18. }
19. $resultantstring = connectWithHyphenAllElementsIn
    (@dehyphenizedwords)
20. change all occurrences of $hyphenatedword in the col-
    lection to $resultantstring
21. }
```

Further explanation of the algorithm is as follows:

For a hyphenated word "paniculi-form-scorpioid," line 4–11 forms a 3×3 square matrix (shaded cells are the cells on the diagonal line):

	Paniculi	Form	Scorpioid
Paniculi (row 0)	[0, 0] = 0 (because paniculi is not a legitimate word)	[0, 1] = 1 (because paniculiform is a legitimate word)	[0, 2] = 0 (because paniculiformscorpioid is not a legitimate word)
Form (row 1)		[1, 1] = 1 (because form is a legitimate word)	[1, 2] = 0 (because formscorpioid is not a legitimate word)
Scorpioid (row 2)			[2, 2] = 1 (because scorpioid is a legitimate word)

Line 12 sorts the rows by the longest distance of a 1 to the diagonal cell on each row, in descending order. For row 1, this distance is the distance between the cell [0, 1] and [0, 0], which is 1; for row 2, the distance is 0 (i.e. the distance between cell [1, 1] and itself); and for row 3, the distance is 0 as well (i.e. the distance between [2, 2] and itself). Because row 1 has the longest distance, the legitimate word "paniculiform" formed on row 1 is saved to @dehyphenatedword on line 15, and the rows representing "paniculi" and "form" (i.e. row 0 and row 1) are removed from further consideration on line 16. This leaves only row 2 to be examined, which contains one legitimate word "scorpioid" (cell [2, 2] = 1), which is also saved to @dehyphenatedword. Connecting all legitimate words in @dehyphenatedword, the resultantstring is "paniculiform-scorpioid" on line 19. Finally all occurrence of "paniculi-form-scorpioid" in the text are replaced by "paniculiform-scorpioid" on line 20.

learnSeedNouns

Function: identify nouns from a description collection.
Exploits: the occurrences of singular, plural, and verb endings of a word in the collection.
Input: a collection of plain text descriptions.
Output: a set of nouns tagged as p (plural) or s (singular).
/*line 1-4: initialize verb endings, noun endings, and special singular and plural noun endings of biological terms, for example "pulvini" is the plural form of "pulvinus"*/

```
1. @verbendings = (ing)
2. @nounendings = (tion, ness, ism, ist, ment, ance, ancy,
   ence, ency, sure);
3. @specialsingularendings = (on, is, ex,ix, um, us, a);
4. @specialpluralendings = (ia, es, ices, i, ae);
5. remove any HTML or other tags from text files
6. read words from files
   /* line 7 to 14 collect words with a special plural/singular
   ending and appearing before "absent" or "present".
```


DOI: 10.1002/asi

```
These words are saved as plural/singular nouns. An example of such case is "Pulvini absent."*/
7. foreach $word{
8. if($word appears before (absent|present) in descriptions and $word has $specialpluralending){
9.     addpluralnoun($word)
10.  }
11. if($word appears before (absent|present) and $word has $specialsingularending){
12.     addsingularnoun($word)
13.  }
14. }
/*line 15–26 look for other nouns by using the following heuristic rule:
A word is a noun if a) its singular and plural forms can be found in the text, and b) its verb form (i.e. –ing) can not be found in the text)*/
15. group words by their roots /* e.g. "abaxial" and "abaxially" are in the same group*/
16. foreach @group{
17.     if (@group contains only one $word and the $word has a $nounending){ /*e.g. if a group contains one word "combination" */
18.         addsingularnoun($word) /*e.g. save "combination" as a singular noun */
19.     }else if(@group has at least two words and none of them has a $verbending ){
        /* e.g. a group such as ("extend," "extends," "extending") does not pass the above test. A group such as ("stem", "stems", "stemless") passes this test */
20.         foreach @pair of words in @group{
        /*a group of x words makes x(x-1)/2 pairs. For example: the group ("stem", "stems", "stemless") gives the following 3 pairs "stem"/"stems", "stem"/"stemless", and "stems"/"stemless" */
21.             ($singular, $plural) = isSingularPluralPair(@pair) /*this function is defined below*/
22.             addsingularnoun($singular)
23.             addpluralnoun($plural)
24.         }
25.     }
26. }
```

isSingularPluralPair(a pair of words): This function decides if a pair of words in the singular and plural forms of a word. It returns yes if (a) one word has a $specialsingularending and the other word has a $specialpluralending and the difference in length between the two words is less than 2, or (b) the longer word contains the shorter word and ends with a combination of letters in [yies] and the difference between the two words in length is less than 3. Otherwise, it returns no.

CoreBootstrapping

Function: assign "n," "b," or "m" roles to words in a collection; annotate individual clauses in a collection. The "n" role may be specified either as "s" (singular) or "p"(plural).
Exploits: the pattern of "modifier-head noun-boundary word" that starts a simple clause.
Input: a collection of text descriptions.
Output: a knowledge base consists of words and their roles; a database consists of clauses annotated with their subject (including the modifier and the head noun).

```
1. extractAndCategorizeClauses() /*Clauses with a plural subject are put in the "start" category. Other clauses are put in "normal" category. */
2. discover ("start")/*clauses in the "start" category is learned first*/
3. discover("normal")/*clauses in the "normal" category is learned after */
```

extractAndCategorizeClauses()

```
1. segment files into clauses by punctuation marks: semicolon (;), colon(:), and period(.)
2. foreach $clause{
3.     if ($clause starts with a plural noun or a word with a plural ending){
4.         save $clause in the "start" category
5.     }else{
6.         save the clause in the "normal" category
7.     }
8. }
```

discover($status)

/$status is either "start" or "normal"*/*

```
1. foreach unannotated $clause with $status{
2. @leadingWords = getLeadingWords($clause, $x)/*get the first $x = 3 words from the clause*/
3. $ptn = buildPattern(@leadingWords, $x)/*see below for buildPattern function*/
4. @matched = getIdsOfUnannotatedClauseThatMatch ($ptn,$status) /*find clauses that match the pattern $ptn*/
   /*line 5 to 11 annotate clauses one by one until there is no new discovery is made in an iteration*/
5.  do{
6.    $madeNewDiscovery = 0;
7.    foreach $id in @matched{
8.      if (the clause with $id is not annotated)
9.        $madeNewDiscovery += dothis($id)     /*dothis annotates the clause whose id = $id*/
10.   }
11.  }while ($madeNewDiscovery > 0)
12. }
```

buildPattern(@leadingWords, $x): returns the pattern that matches any sentences whose first $x words match any word in @words. For example, if @leadingWords = (cat, dogs, fist), then the pattern is /^(cat|dogs|fish)|^\w + \s(cat|dogs|fish)|^\w + \s\w + \s(cat|dogs|fish)/.

dothis($id)

/*this is the core bootstrapping procedure*/

```
1. $new = 0 /*when the program starts, the count of new discoveries is initialized to 0*/
```


DOI: 10.1002/asi

2. $clause = fetchClauseWith($id)
3. $lead = getLeadingWordsFrom ($id, 3) /*get the leading words of a clause. Take no more than 3 words before a punctuation mark as the lead. The lead is likely to contain the subject and the first few boundary words. */
4. $rolePattern = getRolePattern($lead) /*A role pattern is a combination of "p", "s", "n", "b", and "?". "p" stands for plural noun, "s" stands for singular noun, "n" stands for noun (no matter its count), "b" stands for boundary word, and "?" stands for unknown. For example, the role pattern of a lead "flowers red" may be /pb/ if "flowers" is known to be a plural organ name and "red" a boundary. If either of the word is known, the role pattern is /??/ */ /*Line 5 to 50 treats a number of rolepatterns one by one. Not all possible patterns are treated. Only patterns that may provide reliable inferences are treated to avoid errors. Inferences include finding the annotation [tag] for a lead and finding a semantic role for a word. */
5. if ($rolePattern matches /^[pns]$/){ /*e.g. the role pattern of a lead "leaves:" is /p/ */
6. $tag = $lead /*e.g. determine the annotation for the lead and the corresponding clause is "leaves"*/
7. }else if ($rolePattern matches /ps/){ /*e.g. a lead "stylopodia lateral" may has a pattern /ps/ */
8. $tag = all words in $lead up to the p /*annotation = "stylopodia" */
9. $new += change the role s to role b /*identify and correct the error, change the role for "lateral" to "b"*/
10. }else if ($rolePattern matches /p?/){
11. $number = checkWordNetForTheNumberOf(?) /*need to know if the word corresponding to "?" is a plural or a singular to decide how to proceed.*/
12. if($number == p){ /* e.g. "fruits nuts", if WordNet says that "nuts" is a "p", now the pattern is known as /pp/*/
13. $tag = the word corresponding to the 2nd p /*annotation = "nuts"*/
14. $new += updateRole of ? to p /*add "nuts is a p" to the knowledge base, 1 new discovery is made so increase $new by 1*/
15. }else{ /*the pattern remains /p?/. e.g. "flowers solitary"*/
16. $tag = up to the p /*annotation = "flowers"*/
17. $new += updateRole of ? to b /* solitary's role is "b", 1 new discovery is made so increase $new by 1*/
18. }
19. }else if ($rolePattern matches /^sbp/){
20. $tag = all words in $lead up to the p
21. }else if ($rolePattern matches /[psn]b/){
22. $tag = all words in $lead up to the noun before b
23. }else if ($rolePattern matches /[psn][psn]+/ or $rolePattern matches /[?b][?b]([psn])$/){
24. $extendedPattern = $rolePattern+all consecutive nouns from the clause following $lead
    /*patterns end with an organ name (n), but this n may be a modifier for another organ name (e.g. leaf blade). $extendedPattern includes all organ names that were left out from $lead*/
25. if($extendedPattern matches /pp/) {
26. $tag = the word corresponding to the last p
27. $new += updateRole of the word after $extended-Pattern to b
28. }else{
29. $tag = all words represented in the $extended-Pattern
30. $new += updateRole of the word after $extended-Pattern to b
31. }
32. }else if ($rolePattern matches /^s?$/){
33. $number = checkWordNetForTheNumberOf(?) /*check if ? is a plural or a singular*/
34. if($number == 'p'){ //now the pattern is known as /^sp$/
35. $tag = all words represented in $rolePattern
36. $new += updateRole of ? to p
37. }else{ //the pattern remains /^s?$/
38. $tag = the word corresponding to s
39. $new += updateRole of ? to b
40. }
41. }else if ($rolePattern matches /^b[sp]$/){
42. $tag = all words represented in $rolePattern
43. }else if ($rolePattern matches /^?b/){
44. $number = checkWordNetForTheNumberOf(?) /*check if ? is a plural or a singular*/
45. if(($lead does not contain a preposition and is not followed a noun) *and* $number matches [sp]){
46. $tag = word represented by ? /*now the pattern is either /^sb/ or /^pb/ */
47. $new += updateRole of ? to $number
48. }
49. annotateClause($tag, $id) /*annotate the clause with $tag*/
50. return $new; /*$new discoveries have been made. $new == 0, meaning this run has made zero new discoveries, would terminates the learning process. */

leadWordBootstrapping

Function: assign "n", "b" "m" roles to words in a collection; annotate individual clauses in a collection. The "n" role may be specified either as "s" (singular) or "p"(plural).
Exploits: the same organ from different orgnisms may have different boundary words.
Input: clauses extracted from a collection.
Output: new "n", "b", or "m" words; annotations for some previously unannotated clauses.

1. $new = 0 /*when the program starts, the count of new discoveries is initialized to 0*/
   /*Line 2 finds clauses that are un-annotated. Sort them by the length of their "lead", longest lead first. Let's say "stigmatic scar basal" is such an un-annotated clause*/
2. @unannotatedClauses = getUnannotatedClauseWith MultipleWordsLead("SortByLengthOfLead Descendingly");
3. foreach $clause in @unannotatedClauses{ /* e.g. $clause = "stigmatic scar basal, ..." */
4. $lead = getLeadingWordsFrom($clause, 3) /*$lead = "stigmatic scar basal" */
5. $lead = removeLastWordFrom($lead) /*$lead = "stigmatic scar" */
6. if (the last word in $lead is a noun){ /*by checking what has been learned or check WordNet, find "scar" is a noun*/

DOI: 10.1002/asi

7. @clausesWithSameLead = getUnannotatedClauses StartingWith($lead)/*find other un-annotated clauses starting with "stigmatic scar", e.g. find "stigmatic scar apical" */
8. if(@clausesWithSameLead is not empty){
9. annotate the clauses with the shared $lead /*annotate "stigmatic scar apical" with "stigmatic scar" */
10. $new += updateRole of the word immediately following $lead to b /*new discovery: "apical" is a "b", increase $new by 1*/
11. }
12. }
13. }
/*Line 14 to 19 deal with a special case, that is, when a clause contains one single word. Mostly likely the word is an organ name. */
14. @unannotatedClauses = getUnannotatedClauseWith SingleWordLead();
15. foreach $clause in @unannotatedClauses{
16. $lead = getLeadingWordsFrom($clause, 1) /*get the first and the only word*/
17. annotate $clause with $lead
18. $new += updateRole of $lead to n /*increase the count of the new discoveries*/
19. }
20. return $new; /*return the number of new discoveries have been made in this run*/

unknownWordsBootstrapping

Function: infer the role of words whose role is still unknown.
Exploits: the fact that plural nouns, when used as a subject, are the head noun, as opposed to being a modifier of a head noun.
Input: all unknown words, a description collection.
Output: new discoveries on the roles of some previously unknown words.

1. $new = 0;
2. do{
/*Line 3 collects a set of possible plural nouns. Line 5 further tests if a candidate is more likely than not to be a plural noun. If so, further search is done on Line 7 and 9 to find other words that appear immediately after and before the newly discovered plural noun. Those appears before the noun are new "b" words, and those after the noun are new "m" words. */
3. @unknownWordsWithPluralEndings = get all unknown words with a plural ending
4. foreach $unknownWord in @unknownWordsWithPluralEndings {
5. if ($unknownWord is seen to appear before a b word in a description){
6. $new += updateRole of $unknownWord to p /*$new keeps the count of new discoveries*/
7. @discoveredBoundaryWords = find all unknown words seen to immediately follow $unknownWord in the collection
8. $new += updateRole of all words in @discovered-BoundaryWords to b
9. @discoveredModifierWords = find all unknown words seen to immediately proceed $unknownWord in the collection
10. $new += updateRole of all words in @discovered-ModifierWords to m
11. }
12. }
13. }while ($new > 0)

adjectiveSubjectsBootstrapping

1. **Function**: identify and annotate the clauses with an adjective subject.
2. **Exploits**: 1) an adjective subject is followed by a boundary word. 2) "and" and "or" connect words playing the same role.
3. **Input**: all clauses with known words tagged with a role.
4. **Output**: a set of new modifiers; annotations to clauses with adjectiveSubjects.

1. $new = 0
2. do{
/*Line 3 collects possible "adjectives as subject" cases, for example "cauline and <M>basal</M> <B>often</B>" contains a modifier (i.e. basal) followed by a boundary word (i.e. often), which makes "basal" a possible "adjective as subject" case.*/
3. @adjectiveSubjects = findModifiersFollowedByA BoundaryWord();
4. @adjSubClauses = findUnannotatedClausesContaining AnyWordIn(@adjectiveSubjects)
5. foreach $adjSubClause in (@adjSubClauses){
6. if($adjSubClause match /? (and|or) (@adjectiveSubjects) #/){ /*match e.g. "cauline and <M>basal</M>"*/
7. if(# is the end of a clause){ /*e.g. cauline and <basal>;*/
8. $new += updateRole of ? to m /*e.g. cauline is an m*/
9. annotate the clause "ditto"
10. }else if(# is a b){ /*e.g. cauline and <M>basil</M> <B>often. . .*/
11. $new += updateRole of ? to m /*e.g. cauline is an m */
12. annotate the clause with "? and/or $adjectiveSubjects" as the modifier, the parenttag found by looking into the context as the tag /*e.g. tag this clause <cauline and basal [leaves]>*/
13. }
14. }else if($adjSubClause match / (@adjectiveSubjects) b/){ /*e.g. <M>basal</M> <B>often</B> absent*/
15. annotate the clause with the matched $adjectiveSubject as the modifier, the parenttag as the tag /*annotation = <basal [leaves]>*/
16. }
17. }
18. discoverNewModifiers()
19. }while($new > 0)

discoverNewModifiers()

1. $new = 0
2. do{

DOI: 10.1002/asi

```
/*Line 3 and 7 find new modifiers by looking for conjunctions (and/or) between an unknown word and a modifier*/
3.    @segments = findAllSegmentsMatchingPattern( /?(and|or) m b/ ) /* e.g. cauline and <M>basil</M> */
4.    foreach $segment in @segments{
5.      $new += updateRole of ? to m
6.    }
7.    @segments = findAllSegmentsMatchingPattern( /m (and|or) ? b/ ) /*e.g <M>cauline</M> and basil */
8.    foreach $segment in @segments{
9.      $new += updateRole of ? to m
10.   }
11. }while($new > 0)
```


DOI: 10.1002/asi