Generating Domain Terminologies using Root- and Rule-Based Terms[1]

Jacob Collard[1], T. N. Bhat[2], Eswaran Subrahmanian[3,4], Ram D. Sriram[3], John T. Elliot[2], Ursula R. Kattner[2], Carelyn E. Campbell[2], Ira Monarch[4,5]

Independent Consultant, Ithaca, New York[1], Materials Measurement Laboratory, National Institute of Standards and Technology, Gaithersburg, MD[2], Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD [3], Carnegie Mellon University, Pittsburgh, PA[4], Independent Consultant, Pittsburgh, PA[5]

Abstract

Motivated by the need for flexible, intuitive, reusable, and normalized terminology for guiding search and building ontologies, we present a general approach for generating sets of such terminologies from natural language documents. The terms that this approach generates are root- and rule-based terms, generated by a series of rules designed to be flexible, to evolve, and, perhaps most important, to protect against ambiguity and standardize semantically similar but syntactically distinct phrases to a normal form. This approach combines several linguistic and computational methods that can be automated with the help of training sets to quickly and consistently extract normalized terms. We discuss how this can be extended as natural language technologies improve and how the strategy applies to common use-cases such as search, document entry and archiving, and identifying, tracking, and predicting scientific and technological trends.

Keywords

dependency parsing; natural language processing; ontology generation; search; terminology generation; unsupervised learning.

1. Introduction

1.1 Terminologies and Semantic Technologies

Services and applications on the world-wide web, as well as standards defined by the World Wide Web Consortium (W3C), the primary standards organization for the web, have been integrating semantic technologies into the Internet since 2001 (Koivunen and Miller 2001). These technologies have the goal of improving the interoperability of applications on the rapidly growing Internet and creating a comprehensive network of data that goes beyond the unstructured

[1] Commercial products are identified in this article to adequately specify the material. This does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply the materials identified are necessarily the best available for the purpose.
Special thanks to Sarala Padi for assistance in compiling and presenting this document

documents that made up previous generations of the web (Berners-Lee, Hendler, and Lassila 2001; Feigenbaum et al. 2007; Swartz 2013). These semantic technologies can be used to protect against ambiguity and reduce semantically similar but syntactically distinct phrases to normalized forms. Semantically normalized forms allow users to more easily interact with data and developers to reuse data across applications. However, most of these technologies rely on data that have been annotated with semantic information. Data that were not designed for use on the semantic web most likely do not include this information and are therefore more difficult to integrate. For example, scientific research papers are typically text- and graphics-based documents designed to be read and processed by humans. These documents do not usually contain semantic markup, meaning that search engines may not be able to take advantage of such advances in data technologies.

Another major issue in semantic computing is the representation of domain-specific semantics. Different scientific and academic disciplines, as well as other spheres of communication such as conversation, business interactions, and literature, all have overlapping vocabulary. However, the same words often have different meanings depending on the domain. In the sciences, each field (and often each subfield) has its own terminology that is not used in other disciplines, or that conflicts with the language of more general-purpose communication. For example, in general use, the word *fluid* typically includes liquids, but not gases. However, in physics, *gas* and *liquid* are both hyponyms of *fluid*. Semantic technologies may assume that two annotations with the same name have the same semantics, when this is not necessarily the case.

Generally speaking, this issue stems from the problem of coordination described by (Clark and Wilkes-Gibbs 1986). Any participant in a system of communication is typically missing some of the knowledge held by other participants. Because of this knowledge gap, participants may not understand one another unless they are using a shared knowledge system and a shared vocabulary. (Clark and Wilkes-Gibbs 1986) describe how speakers establish a **common ground**, collaborating to ensure that participants know one another’s strengths and limitations. Interactions involving computers are also systems of communication, and must also coordinate in order to ensure that all applications are communicating properly. Establishing common ground across fields is supported by standardizing terminology and having hyponymic and other semantic relations structured for use by humans and machines.

Issues of coordination are relevant in many fields, particularly when it comes to data re-use and interoperability. For example, The Minerals, Metals, & Materials Society (TMS) describes many gaps and limitations in current materials science standards; one of these gaps is an “insufficient number of open data repositories,” referring to repositories containing data that can be used by many applications, with the stipulation that data not only be available, but also be re-usable (“Modeling Across Scales” 2015). This is impossible without some means of coordination and standardization of terminology. TMS recommends developing initiatives to aid in coordination, for example by engaging “a multidisciplinary group of researchers to define terminology and build bridges across disciplines.” Multidisciplinary coordination is a necessary part of improving data reusability, but support of such coordination is lacking.

Our goal in this paper is to describe a general system that is capable of automatically creating standardized terminologies that will be useful for developing domain ontologies and other data structures to fill this gap. A domain ontology is a collection of concepts (represented as terms) and relations between them that correspond to knowledge about a particular family of topics. Our system will take into account potential issues in terminology generation, including the disambiguation and normalization of terms in a domain. In many cases, a single term can be expressed in many different ways in natural language. For example, in mathematics the phrase “without loss of generality” has a technical meaning; however, a researcher might also write “without any loss of generality” or “without losing generality.” All three variants refer to the same technical phrase and should be treated together in a standard terminology. Terminologies also face issues relating to polysemy, syntactic ambiguity, context sensitivity, and noisy data.

The key component of our system is a representation of terminology that takes advantage of the compositional nature of natural language semantics by converting natural language phrases into consistently structured terms. This representation overcomes issues of syntactic variation by normalizing different syntactic structures based on their compositional semantics. That is, we represent synonymous phrases in the same way despite differences in surface realization. To help ensure consistent terminology generation, our system uses a set of rules to restrict and guide the formation of these normalized terms. Because this system is based on a modular set of rules that combine smaller components of phrases into standardized terms, we refer to it as a **root- and rule-based method** for terminology generation.

Our root- and rule-based approach is motivated more by linguistic than statistical models. This is necessary for the construction of rule-based terms, which are dependent on consistent structures and a representation of the underlying linguistic form. Our rule-based model ultimately relies on the way that words come together syntactically in order to form phrases. The meanings of these phrases are, in most cases, related to the meanings of their component parts (i.e. the individual words). The way that words compose to form more complex meanings is detailed in research such as (Montague 1988), though the underlying principles ultimately extend back to (Frege 1884). Through an understanding of syntax as modeled in linguistics, it is possible to formalize the compositionality and therefore normalize synonymous phrases, despite significant differences in form.

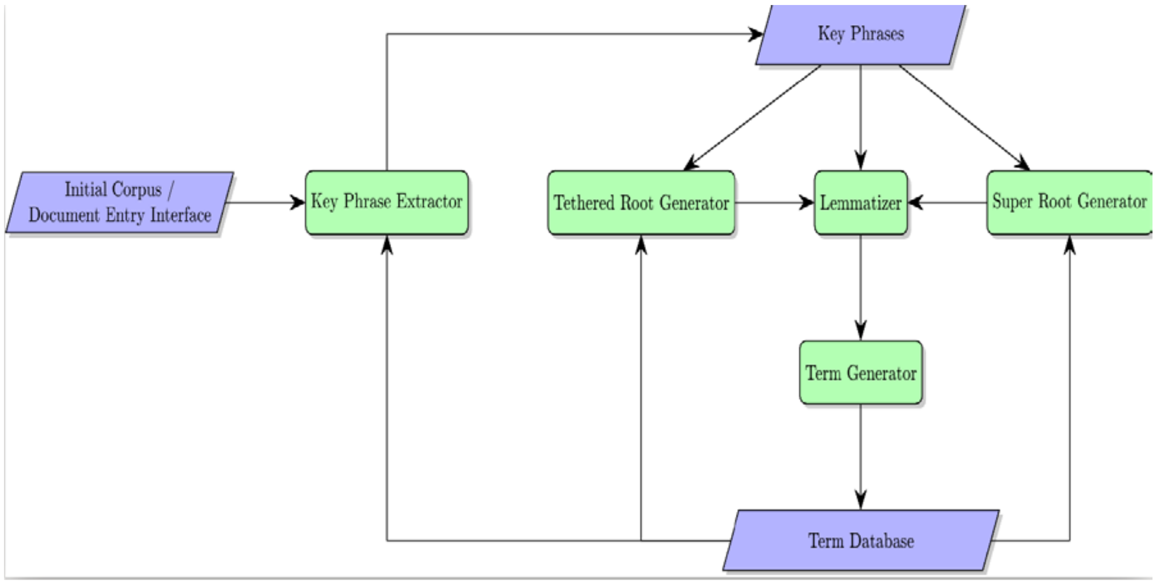

Figure 1. Terminology Generation

Within the context of our root- and rule-based system, a **term** is a representation of a concept within a domain (and may cover a number of words and/or phrases); a collection of terms describing the same domain make up a **terminology**. Our system also defines **roots**, which are smaller components which come together to make up terms – that is, a single term is made up of one or more roots. A **terminology** is distinct from an **ontology** in that an ontology additionally defines the relationships between concepts, though the concepts in an ontology are typically represented by terms of some sort. A **rule** is a codified process used to generate, restrict, or normalize terms in our root- and rule-based approach. This paper will describe the linguistic and theoretical motivations behind these rules, which are introduced in (Bhat et al. 2015) as specifically applied to materials science. In Section 2, we describe the theory behind the syntactic normalization that allows for the automatic construction of root- and rule-based terms. This is followed by Section 3, where we describe how to use features of natural language syntax in combination with additional rules to create root- and rule-based terms. We then describe how terms can be extracted from natural language texts in Section 4. One of the main features of the root- and rule-based approach is that it is easily extensible and adaptable to new and different use-cases, as we discuss in Section 5. Lastly, we describe how our system ties in with the challenge described above of creating terminologies that are robust to the complexities of natural language and to the needs of users in Section 6.

1.2 Use-Cases and Architecture

In developing this strategy, we have considered four very general use-cases, each of which corresponds with an interface that allows users and administrators to interact with the terminology generation system. These will be discussed in greater detail in Section 6. We also discuss ontology generation as an extension of terminologies generated from the root- and rule-based method.

- **Document Entry:** Users should be able to upload documents, have terms extracted from these documents, make changes to the suggested terms, and have the terms added to the terminology in the database.
- **Document Retrieval:** Users should be able to construct a query and receive a list of documents matching the terms in the query.
- **Curation:** Curators (who may be dedicated administrators or user volunteers in crowd-sourced systems) should be able to make changes to the terminology.
- **Rule Changes:** Curators should be able to make changes to the way the system generates terms, as technologies change. Changes also reflect the way that people are using the terminology and systems that have become *de facto* standards.
- **Ontology Generation:** Users should be able to use a set of terms as the basis for a domain ontology, which extends a terminology by providing additional semantic relationships between terms.

With these in mind, we have outlined a root- and rule-based terminology system in Figure 1. This figure shows the general process which will be explained in this paper. Through this process, a set of rules are used to extract a series of key phrases from a corpus and convert them into root-based terms (three types of roots are described in this paper: roots, tethered roots, and super-roots; see Section 3).

2. Theory

Generating a terminology requires identifying salient concepts in a corpus and constructing representations of those concepts. There are many different approaches to identifying salient concepts, that is, to find words and/or phrases in the text that stand out. In many cases, the identification of salient concepts produces a list of words and phrases taken directly from the text. A terminology generation system then needs to convert words and phrases into a format which enhances the potential for humans and machines to use the terminology for various practical applications. As with key phrase extraction, researchers and developers have used a variety of methods to convert words and phrases into terms representative of key concepts (Witschel 2005).

2.1 Previous Research

Key phrase selection is a major area of study in the field of information extraction (Witschel 2005). Many methods of key phrase extraction rely on two components: a unithood metric and a termhood metric. A unithood metric determines the particular types of words and phrases that qualify as key phrase candidates. Units found by a unithood metric may or may not be relevant enough to qualify as key phrases, but can be used to restrict the set of words that must be compared for relevance. For example, a unithood metric may identify all noun phrases in a corpus, so that only noun phrases are considered as potential terms. Not all of the noun phrases selected will become terms, but only noun phrases will be extracted. More complex unithood metrics are also possible. (Frantzi, Ananiadou, and Tsujii 1998), for example, consider the following unithood metrics in the evaluation of their C-Value and NC-Value algorithms, which are algorithms for key phrase extraction[2]:

- Noun$^+$Noun
- (Adj|Noun)$^+$Noun
- ((Adj|Noun)$^+$|((Adj|Noun)*(Noun Prep)$^?$)(Adj|Noun)*)Noun

A typical automatic term recognition algorithm may then identify which of the selected units are relevant using a termhood metric. A termhood metric may be based on statistical or linguistic features; (Frantzi, Ananiadou, and Tsujii 1998) use word frequency to identify nested terms (candidates which occur within other candidates) and the surrounding context of a term. These are used together with a mathematical formula to assign a score to each candidate. In this way, they are able to select for particular types of terms. The features used by (Frantzi, Ananiadou, and Tsujii

[2] In this representation, *Noun*, *Adj*, and *Prep* are patterns matching parts of speech (noun, adjective, and preposition, respectively). Parentheses group patterns together, and two patterns separated by a pipe (j) produce a new pattern which matches either of its components. The asterisk (*) produces a pattern that matches the previous expression zero or more times. The plus sign (+) is similar, but matches the previous expression one or more times, while the question mark (?) matches the previous expression zero or one times.

1998) are by no means exhaustive; (Proux et al. 1998) and (Rindflesch, Hunter, and Aronson 1999), for example, make use of linguistic information such as part-of-speech tags to improve their termhood metric.

We do not present any new methods for automatic term recognition, nor do we make any judgment as to the “best” contemporary method. However, because the model of term generation and normalization that we describe is dependent on automatic term recognition, we do discuss it briefly. In theory, our system can be used with any automatic term recognition algorithm, though multi-word terms, such as those recognized by the C/NC-Value Algorithm (Frantzi, Ananiadou, and Tsujii 1998) are optimal for the root- and rule-based method as hierarchical relationships can be inferred from these complex terms.

Once terms have been selected, we generate normalized concepts rather than using natural language phrases. Natural language phrases have many disadvantages, including ambiguity and synonymy, as discussed previously. One method of normalizing concepts is to automatically group words into clusters, as in (Liu et al. 2012). For example, the phrases “monthly expense,” “personal insurance product,” “core product,” “voluntary benefit,” and “personal insurance” may all be clustered to form a single concept representation related to insurance. In this way, word choice among different authors and contexts is normalized – words whose appearance is positively correlated are grouped together. The disadvantage of clusters is that they are difficult to label – other than appearing as the set of words in a given cluster, they are not human-readable.

Other approaches to normalization may involve various other statistical and natural-language processing techniques, such as (Park, Byrd, and Boguraev 2002), who use a combination of stop word removal, lemmatization (normalization of different forms of a word, .e.g, *people* and *person* or *colors* and *color*), and abbreviation detection. There are various components of a key phrase that may not be desirable in terminology generation. Inflectional morphology – grammatical affixes such as *-s* and *-ing* provide grammatical information within the context of a natural language sentence, does not usually differentiate between technical terms. A terminology should not usually extract both *heat capacity* and *heat capacities* – these are probably both instances of the same term. Certain functional words, including articles such as *a*, *an*, and *the* may also be unhelpful. These issues can be dealt with through lemmatization and stop word removal, both of which are well-known problems with many proposed solutions in the field of natural language processing (Park, Byrd, and Boguraev 2002). However, even assuming that we can lemmatize phrases and remove stop words, there may still be undesirable redundancies in an automatically generated terminology, as there are many ways to express the same thing by using different syntactic structures.

Consider, for example, the syntactically similar phrases *the red tree leaves* and *the red leaves of the tree*. In many contexts, these phrases have approximately the same meaning – they both refer to leaves which belong or grow on a tree and are red in color. When dealing with phrases such as these, a terminology extraction algorithm should be able to reduce redundancy by converting both of these terms into a normalized syntactic pattern. A great deal of theoretical and applied linguistic and computational research has been done regarding the determination of a sentence or phrase’s syntactic structure. By applying *dependency parsing* to terminology extraction, it becomes possible to normalize the syntactic structure of phrases. While many other terminology generation algorithms work with phrases, one of the main benefits of our approach is the ability to normalize surface-level differences in the structure of these phrases.

2.2 Dependency Grammar

As mentioned above, we use dependency parsing to normalize the syntactic structure of phrases. A dependency parser is a tool for syntactic analysis that produces a dependency tree, which is defined in terms of the relationship between a phrasal head and the rest of the phrase (Tesnière 1959). The phrasal head carries the syntactic category of the phrase – e.g., a noun phrase is headed by a noun, a verb phrase by a verb, etc. Furthermore, a dependency represents some semantic information, as the head of a phrase is typically specified by the rest of the phrase. For example, the phrase *the tree's red leaves* (a noun phrase) is headed by the word *leaves*. The word *leaves* on its own is quite general – it could refer to the leaves of any plant whatsoever. However, because of the rest of the phrase, we know that the leaves in question are red and that they belong to or come from a tree.

Dependency syntax can be represented using a tree structure such as that in Figure 2. Modern parsing technologies, such as the Stanford Parser (Manning et al. 2014) and MaltParser (Hall 2006) are capable of automatically generating dependency trees from text. With the help of these tools, we can take advantage of the semantic information provided by syntactic structures and normalize the syntax of phrases to generate terms.

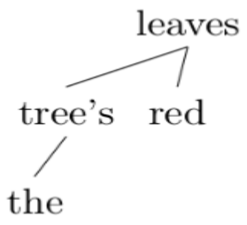

Figure 2: *Dependency representation of the tree's red leaves*

3. Representing Root- and Rule-Based Terms

3.1 Guidelines for Terminology Representation

Because the goal of providing a domain terminology is to produce a list of formal concepts in the domain, it is necessary that generated terms be both unambiguous and relevant to the domain. If terms are ambiguous, then the terminology will be inaccurate. If terms are irrelevant, even if their semantics are correctly represented, the terminology will not be of any practical use. We have defined the following criteria to describe useful terms for domain terminologies, based on rules 1 to 10in (Bhat et al. 2015) (reproduced in Section 7) These criteria should generally apply to any terminology generation schema.

- Terms should be human-readable and machine-friendly. All terms should be based on natural language.
- The same term representation should always identify the same concept within a terminology; similarly, two terms with different representations should represent different concepts (see (Bhat et al. 2015), rules 5,7, and 8).

- The meaning and form of a term should be predictable from smaller parts. This predictability must be applicable to both humans and machines (rules 8,9, and 10).
- The form of a term should be predictable – that is, given a particular meaning, it should be possible to derive a compositional name for a term with that meaning (rules 2,3, and 6).
- Given a term's compositionality, both humans and machines should be able to identify semantic relationships.
- Terms should be intuitive enough that both humans and machines can identify existing semantic relationships between them.
- Terms should be mutable enough that new terms with related semantics can be generated (rules 3, 4, and 8)
- Only terms representing discriminating concepts should be generated (rules 2 and 8).
- Terms representing highly specific instances and individuals should generally not be generated; terms should be reusable in many use-cases (rule 9).

Generally speaking, terms will represent a hierarchy, with some concepts being more specific than others. Thus, there is no concrete definition for "too general" or "too specific" as applied to a terminology; we are simply looking for terms that are useful at the level of specification provided by the domain, keeping in mind that terms should be usable for data representation, sharing, and analysis. The domain terminology should be representative of the input corpus.

3.2 Term Representation

3.3 Normalized Dependency Trees

The term representation that we have developed takes advantage of common features of natural language to create a human-readable schema that maintains the stipulations given above (though building more complex structures, such as ontologies, is also dependent on other compounds, such as the term selection strategies discussed in Section 4) and Section 8.

The theoretical framework for our term representation is the syntactic structure of phrases and dependency grammar, as discussed in subsection 2.2. Though dependency trees are not easily human-readable, they can be converted into human-readable forms, due to one important feature: given a dependency parse, it is not the order of the nodes which determines the meaning of the phrase – all isomorphic trees have the same meaning. The hierarchical structure of a dependency tree represents all of the semantic information that our strategy relies on; the linear order of the daughter nodes represents only the surface ordering of morphemes and does not on its own contain any semantic information. That is, the trees shown in Figure 3 and Figure 4 are semantically very similar, despite differences in node order. Automatic dependency parsers, such as MaltParser (Hall 2006), will produce similar trees, though we have filtered function words from these in order to improve normalization.

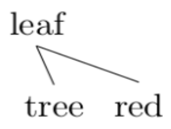

Figure 3: Collapsed representation of the leaves of the tree that are red

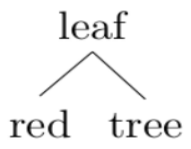

Figure 4: Collapsed representation of the red leaves of the tree

Because of this fact we can re-order the nodes in a set of trees such that all trees follow the same pattern. In this way, trees that are different, but represent the same concept, will generally be normalized to the same structure. For example, if both Figure 3 and Figure 4 were changed such that all nodes were exclusively left-branching (i.e. in a form such that the daughters of a node all appear to the left of the node), they would be identical. This creates a normalized dependency tree that we use to create normalized representations of terminology.

Creating these normalized representations from key phrases is a three-step process: first, a dependency parser creates a dependency tree for each input phrase. Second, a filter removes all function words and other stop words such as prepositions from the dependency trees. Third, each dependency tree is made entirely left-branching.

Though these normalized trees are good representations of phrasal semantics, they are not easily understood by human users of a terminology. Understanding these trees requires an understanding of dependency grammar, a potentially non-intuitive concept. Instead, our strategy converts these normalized trees into human-readable forms using a number of concrete rules. These new representations are linear and can be stored as simple strings of characters.

For example, the trees in Figure 3 and Figure 4 can be linearized by first converting the trees into normalized form (Figure 5). Once we have normalized the syntax, we can convert the structure into a linear format, such as RED-TREE_LEAF. This format contains the same information as the tree structure, and is easily interpreted by English speakers and by computers. For English speakers, the linear form corresponds to a standard English phrase with the same meaning (*the red tree leaf*). For computers, the underscore ('_') indicates that the two final roots (*tree* and *leaf*) compose first, followed by *red*, which is equivalent to the structural information shown in the tree. We have not yet discussed exactly how this linear format is reached from the dependency tree; this, including the semantics of the hyphen and underscore delimiters, will be explained in the following section.

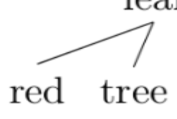

Figure 5: Normalized tree for RED-TREE_LEAF

3.2.2 Roots and Terms

Just as the above structural representations depend on the principle of compositionality (Frege 1884) and the formulations of compositional semantics (Montague 1988), the linear term representation that we have developed facilitates breaking terms apart into smaller components. Unlike in natural language, we primarily use compounding, combining individual “words” into larger terms. The individual meaningful components of a term are called **roots**, and cannot usually be broken down into further meaningful parts. A root should correspond to a single meaningful word such as *tree*, *electron*, or *computational*. Roots can be combined in various ways to create **structured terms**, which are semantically complex structures whose meaning can be easily determined from their component parts. This is based on rules 1, 2, and 3, as well as the specialized terminology in (Bhat et al. 2015).

Roots can combine in different ways; each method of combination is represented textually by a unique delimiter, including the underscore (‘_’), the hyphen (‘-’), and the colon (‘:’). This delimiter notation is extensible and replaceable (completely different sets of delimiters may be used); new delimiters can be added to express new relationships between roots. The hyphen is a general delimiter used for combining roots into terms. Depending on the use-case of the terminology, the hyphen may have a slightly different meaning, but it should usually be used to add specificity to a root through the addition of a second root. For example, the term TREE-LEAF is composed of the roots TREE and LEAF. The hyphen indicates that the root *leaf* (a very general term) is made more specific through the addition of the root TREE, which indicates that the term as a whole represents a specific type of LEAF – namely, the leaf of a tree, rather than the leaf of a bush or other plant. Forms such as this can be easily derived from syntactic structure: both the construction of terms from roots and the structure of syntactic dependency indicate the modifier-head relationship between two roots. In dependency structures, the dependencies of a root are its modifiers, just as a root (the head) is modified by the preceding root.

The interpretation of terms with three or more roots could be ambiguous. However, we impose a left-branching dependency syntax on all terms, meaning that the roots in a term compose from left to right. For example, the term OAK-TREE-LEAF refers to the leaf of an OAK-TREE, and OAK-TREE refers to a tree of type oak. The first semantic operation is the composition of OAK and TREE. This is followed by the composition of OAK-TREE (as a single concept) and LEAF.

Combining roots with more complex structures requires additional delimiters. For example, as described above, the term RED-TREE-LEAF refers to the leaf of a red tree, not to the red leaf of a tree – that is, in this term, the tree is red but the leaf is not. The English phrase “red tree leaf” is ambiguous in a way that the term RED-TREE_LEAF is not. This is why, if we are trying to create a term with the interpretation *the red leaf of a tree* (the dependency structure in Figure 5), it cannot be represented using hyphens alone. The second delimiter, the underscore (‘_’) has higher precedence in the compositional order of operations – composition of roots delimited by underscores occurs before the composition of roots delimited by hyphens. The latter has lower precedence in decomposition. Two roots delimited by an underscore are called “super-roots” and allow for terms that cannot be expressed solely by hyphens. For example, “the red leaf of a tree” (Figure 5) can be represented as the term RED-TREE_LEAF. TREE_LEAF is a super-root and thus combines first: the term represents the leaf of a tree. The super-root then combines with RED, specifying that the TREE-LEAF is of the color red.

We define two additional methods for combining roots, though more methods can be added based on use-case (see Figure 5). The first is to combine the roots without any delimiter at all, referred to as creating a "tethered root." For example, if TREE and LEAF were to be combined into a tethered root, they would become TREELEAF. The purpose of tethered roots is to create roots that are composed of multiple words in English, but which have meaning only as a whole phrase. If the components of a tethered root are represented as roots in a term, the meaning of the term will not follow from its component parts. Set phrases such as "gray area" (referring to a situation that does not easily fit into preexisting categories) are strong candidates for tethered roots. More generally, tethered roots are useful when the component parts are not usable the same way in other terms. For example, if GRAY-AREA is treated as a term, there should be other terms of the form GRAY-X where "gray" has the same meaning as in GRAY-AREA. However, because this is not possible, it is preferable to create a tethered root: GRAYAREA.

The last delimiter we describe in this paper is the colon (':'). Two or more terms can be combined into **compounded terms** with this delimiter. A compounded term represents a high-level semantic cluster, though the usage of these terms is dependent on the use-case. While most terms represent concepts, compounded terms can also represent relationships between those concepts. For example, because APPLE-TREE and RASPBERRY-PLANT both refer to plants that bear fruit, the compounded term APPLE-TREE:RASPBERRY-PLANT could represent this fact about the two component terms. The exact type of relationship is not specified by the compounded term, which describes only a very general semantic connection between its components.

All of these combinations can be generated automatically from a list of key phrases using dependency parsing and a training set. Roots are usually equivalent to nodes in a dependency structure. Because of this, they can be combined into super-roots and into terms by examining an automatically generated dependency tree, removing unimportant words, and performing automatic lemmatization. Creating tethered roots and compounded terms cannot be done with dependency trees alone, and requires the use of a training set. Tethered roots are formed when splitting the term does not provide any reusable information. This can be measured using statistics such as term frequency-inverse document frequency (a measure of a words importance in a document relative to its overall frequency) (Wu et al. 2008). Compounded terms can be identified using measurements of co-occurrence frequency, which identify semantic relationships between terms (Kostoff 1993). Because terms are unambiguous, and different relationships between roots are represented by different delimiters, a machine can also easily break down a term into its component parts, just as it can build up a term based on the relationships between the components (Table 1).

Table 1. Summary of term syntax

Root Type (delimiter)	Description	Example
Term (-)	Composite Concept	OAK-TREE
Root	Single Concept	TREE
Super-Root	High-precedence Composite	TREE_LEAF
Tethered Root	Multi-word Single Concept	GRAYAREA
Compounded Term (:)	Two related terms	APPLE-TREE:RASPBERRY- PLANT

4. Key Phrase Extraction for Root- and Rule-Based Terms

In Sections 2, 3, and 4, we have discussed how to generate structured terms using a root- and rule-based approach taking advantage of syntactic, semantic, and statistical cues. Though structured terms are useful representations of concepts in a terminology, and though natural language processing and other statistical tools can convert key natural language phrases into structured terms, we have not yet discussed how these key phrases are selected. It is possible, of course, to manually select key phrases to be used in a terminology. In some cases, this is unavoidable, as there will always be disagreement as to what constitutes an important term within a domain, but it is helpful if at least some of the work can be done with an automated system. The automated system may be helpful in providing an empirical basis for coming to agreement.

The study of automatic terminology extraction is a major area in the field of information extraction (Witschel 2005). Most methods of terminology extraction rely on two components: a unithood metric and a termhood metric. A unithood metric determines the particular types of words and phrases that make potential candidates for terms. A unit is not necessarily the final representation of a term, nor are all units relevant enough to be treated as terms. For example, a unithood metric might consider all noun phrases (such as “the red leaf of a tree”) in a corpus to be valid term candidates. The task of a termhood metric is to determine which of the candidate terms are important enough within a document to be a part of the terminology. Together, a unithood metric and a termhood metric can extract all of the salient words and/or phrases from a document.

Most terminology extraction methods combine statistical and formal methods. Unithood metrics are often based partially on linguistic features such as part of speech. For example, it is uncommon to include isolated prepositional phrases in a unithood metric (though prepositional phrases that are included in other phrasal categories may be included). Termhood metrics usually analyze the frequency of a term in a document with respect to its frequency in a collection of documents in order to determine the extent to which the term represents the content of the document. However, termhood metrics can also take into account linguistic features; for example, nouns with Greek or Latin endings such as */itis* or */scopy* may be more likely to be technical terms in certain domains (Witschel 2005).

The proposed methods of syntactic analysis described in Section 2 can take as input a series of phrases extracted from a document or corpus and convert them into structured terms. However, this algorithm is sensitive to the terminology extraction techniques used, as it is dependent on the interface between syntax and semantics. If the input phrases are too short to provide semantic clarity, the output terminology will be too general. If the input phrases are too long (with respect to the number of words), the output terminology will be too specific. Many terminology extraction algorithms only extract single morphemes or words – that is, they output terms such as “solar” or “photovoltaic” (Witschel 2005). However, our proposed system prefers units with two to five content words, as longer or shorter terms will tend to be either too general or too specific for most use-cases. Longer terms are more specific, and will often introduce nuances that are not necessary in domain terminologies.

Bearing these restrictions in mind, there are still terminology extraction algorithms that cater to the needs of structured terminologies. Methods such as the NC-Value Algorithm (Frantzi, Ananiadou, and Tsujii 1998) are designed for extracting multi-word terms and their algorithm can easily be extended to favor two- to five-word phrases in order to generate the most effective structured terms.

Though our proposed system is sensitive to terminology extraction, the exact algorithm used is an implementation detail that can be changed easily, as discussed in Section 5. Different use cases may choose different terminology extraction algorithms, depending on their needs. The root- and rule-based approach that we describe is not specific to any terminology extraction algorithm, and the exact method can be customized according to the use case.

The root- and rule-based approach we propose also provides a significant advantage for terminology extraction. Because root-based terms can easily be broken down into their component pieces, it is possible to compare two terms and find similarities between them. Because of this, it is possible to use previously generated normalized terms as hints for term selection. For example, given that the term RED-TREE_LEAF is salient in a corpus, the terms RED-TREE_BRANCH, GREEN-TREE_LEAF, and RED-BUSH_LEAF are probably salient as well, as they share much of the same information.

5. Extensibility

The previous sections describe the various methods that go into terminology generation. The major components are salient phrase extraction (Section 4) and converting key phrases to structured terms (Section 2 and Section 3). However, using this model in a complete system is more complex.

One of the primary benefits of this root- and rule-based approach is the compositional form of terms. Based on this approach, it is possible to build an extensible and modular system that can be adjusted to suit different needs. In this section, we describe how the system as a whole can be configured through different modules and extensions, and how this is enabled by the rule-based model.

Figure 1 shows the various processes involved in generating terms from a corpus of documents. These processes interact with three different types of data: the corpus, the set of key phrases, and the terminology (stored in the database). The corpus may be either a large set of documents used to initialize the terminology, or a smaller set of documents introduced through a user interface, as discussed in Section 6. The set of key phrases are the salient phrases extracted from the corpus; key phrases are natural language phrases that have not yet been processed by the structuring methods described in Sections 3 and 4. The terminology consists of any pre-generated terms, which can be used as a training set. During the term generation process, new terms are added to the existing terminology.

Working with these data requires many different tools and sub-processes - key phrase extraction, tethered root generation, super root generation, lemmatization, and term generation. A key feature of the design is that these tools are not necessarily co-dependent, and can thus easily be substituted depending on the needs of users or on advancements in the technologies that constitute each component.

5.1 Key Phrase Extractor

The key phrase extractor has the task of extracting salient phrases from the corpus. As discussed in Section 4, there are many different algorithms that can handle this task. As such, it is possible to entirely replace the key phrase extractor that is used in a root- and rule-based terminology generation system.

The key phrase extractor may also take advantage of any terms already in the term database by treating them as a training set. Different terminology extraction algorithms may use this training data in different ways. For example, some applications may only wish to include very close matches with preexisting terms, while others may choose to be more liberal with key phrase extraction. This allows different use-cases to use the preexisting terms as appropriate.

5.2 Tethered Root Generator

Tethered roots (see Section 3) may be generated based on two major components of the terminology generation system: the set of key phrases, and the set of preexisting terms. A tethered root generator may use statistical models to determine the information content of a given root relative to the set of all key phrases, or it may use other tethered roots in preexisting terms as cues to generate tethered roots from the current corpus. Again, this component can be customized according to the use-case. One possibility is using Shannon entropy (Shannon 1948) to identify sequences that add less information to a dataset when split up into multiple roots than they would if a tethered root were used instead. For example, if OAK-TREE-LEAF provides less information than OAKTREE-LEAF, then OAKTREE-LEAF could be used instead. Ideally, a tethered root generator will consider not only how much information is contained in each variant, but also whether the information is misleading or inconsistent.

5.3 Super Root Generator

Super root generation requires much of the same information as tethered root generation, except that roots are more likely to make sense when considered separately. Super root generation may also take advantage of a dependency parser in order to determine super roots based on syntactic structure.

5.4 Lemmatizer

In order to avoid creating unnecessary terms, roots are lemmatized to avoid codifying differences in grammatical form. There are many different ways that words can be lemmatized, and lemmatization is a non-trivial task in computational linguistics (Sharma 2010). One common method is to use lexical databases such as WordNet (Fellbaum 1998), as in our forthcoming reference implementation of root- and rule-based terminology generation.

5.5 Term Generator

A term generator combines the roots that make up each key phrase into a structured term based on the results of a dependency parser and the methods described in Section 2 and Section 3. The rules in the rule-based system we describe are not static and can be changed by users, administrators, or developers when needed to improve the system’s performance at its given task.

Altogether, these tools and sub-processes come together to form a model of terminology generation that is customizable, takes advantage of both linguistic and statistical facts, and is at the same time both machine- and user-accessible.

5.6 Example

An example of the entire terminology generation process is shown below, to illustrate how these components come together. This example begins with the following document, taken from (Overton Jr and Gaffney 1955) (https://materialsdata.nist.gov/dspace/xmlui/handle/11256/79), from which terms will be extracted.

> The ultrasonic pulse technique has been used in conjunction with a specially devised cryogenic technique to measure the velocities of 10-Mc/sec acoustic waves in copper single crystals in the range from 4.2K to 300K. The values and the temperature variations of the elastic constants have been determined. The room temperature elastic constants were found to agree well with those of other experimental works. Fuchs' theoretical c44 at 0K is 10 percent larger than our observed value but his theoretical c11, c12, K and (c11–c12) agree well with the observations. The isotropy, (c11–c12)2c44, was observed to remain practically constant from 4.2K to 180K, then to diminish gradually at higher temperatures. Some general features of the temperature variations of elastic constants are discussed.

A key phrase extractor then determines the most salient phrases in the corpus and extracts them. Key phrase are shown in red and underlined.

> The **ultrasonic pulse technique** has been used in conjunction with a specially devised cryogenic technique to measure the velocities of 10-Mc/sec acoustic waves in **copper single crystals** in the range from 4.2K to 300K. The values and the **temperature variations of the elastic constants** have been determined. The room temperature elastic constants were found to agree well with those of other experimental works. Fuchs' theoretical c44 at 0K is 10 percent larger than our observed value but his theoretical c11, c12, K and (c11–c12) agree well with the observations. The isotropy, (c11–c12)2c44, was observed to remain practically constant from 4.2K to 180K, then to diminish gradually at higher temperatures. Some general features of the temperature variations of elastic constants are discussed.

Once these key phrases have been extracted, they need to be converted into normalized root- and rule-based terms. This involves parsing, lemmatizing, and structuring the phrases according to the methods discussed in Sections 2 and 3.

For example, the phrase *temperature variations of the elastic constants* should be parsed and converted into the following dependency tree (Figure 6):

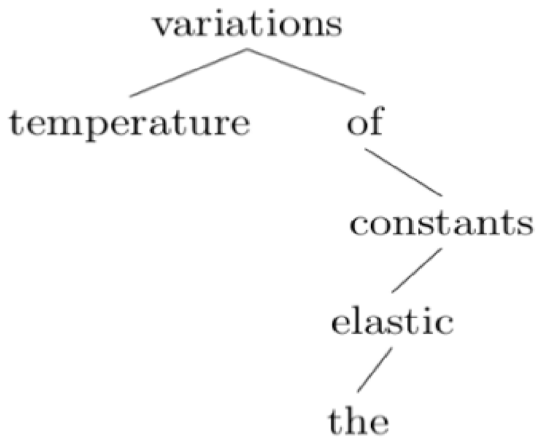

Figure 6: Dependency representation of temperature variations of the elastic constants

The words are then lemmatized and the syntactic structure normalized such that the tree is entirely left-branching. At this time, unimportant function words such as *of* are also removed. This results in Figure 7.

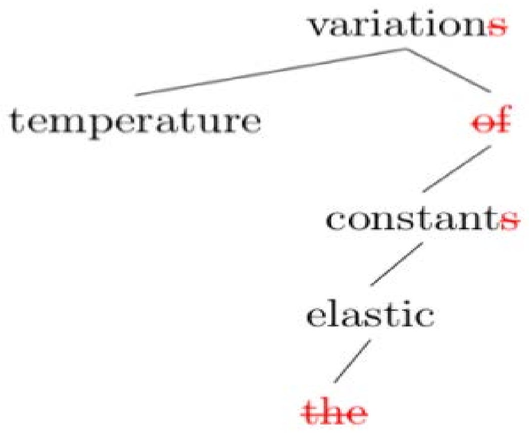

Figure 7: Normalized representation of temperature variations of the elastic constants

Based on this structure, the system should generate the term ELASTIC-CONSTANT-TEMPERATURE_VARIATION. Because *variation* has two branches in Figure 6, one of them must be used to generate a super-root in order to preserve unambiguity. This yields TEMPERATURE_VARIATION which can then compose normally to create the complete term ELASTIC-CONSTANT-TEMPERATURE_VARIATION. The other key phrases can be put through this same process, yielding ULTRASONIC-PULSE-TECHNIQUE, COPPER-SINGLE_CRYSTAL, ELASTIC-CONSTANT-TEMPERATURE_VARIATION, and ROOM-TEMPERATURE-ELASTIC_CONSTANT, which are inserted into the database as valid terms. Some discrepancies may result from this strategy; for example, the above terms include both the structures ELASTIC-CONSTANT and ELASTIC_CONSTANT. Such ambiguities are resolved through the use of a training set or manual curation. For example, if ELASTIC_CONSTANT is found in the training set, the system can resolve this conflict.

5.7 Performance and Usage of Root- and Rule-Based Terms

The previous sections demonstrate how root- and rule-based terms can be constructed using phrases extracted from natural language texts. However, we have yet to analyze the performance of root- and rule-based terms as data structures. One of the major advantages of these structures is that they are capable of being constructed automatically using linguistic and statistical methods, but the structure of the terms themselves provides additional performance and usability gains for many tasks.

The example given in Section 5.6 shows how a small set of root- and rule-based terms can be generated from a single document. If this process is repeated over a larger sample of documents, the result is a large terminology representing concepts in a particular domain. In order to analyze this terminology, we examine the data that are represented by each term.

To begin with, root- and rule-based terms are typically human-readable and understandable. The terms COPPER-SINGLE_CRYSTAL and ULTRASONIC-PULSE-TECHNIQUE are fairly simple to understand, assuming that the component parts are understood. Even without knowing what *ultrasonic* means, it is still possible to gather that ULTRASONIC-PULSE-TECHNIQUE refers to a technique involving ultrasonic pulses. To a human, root- and rule-based terms may not always seem unambiguous - ELASTIC_CONSTANT-TEMPERATURE_VARIATION may initially be perceived as referring to something of constant temperature rather than to the temperature variations of the elastic constant. However, the correct interpretation is typically apparent in English-language terms, as English tends to follow a head-final structure[3] (which is imposed on all root- and rule-based terms).

From a computational perspective, root- and rule-based terms are syntactically unambiguous. The term ELASTIC_CONSTANT-TEMPERATURE_VARIATION refers specifically to the temperature variation of the elastic constant; it cannot refer to elastic variations of constant temperature or to any other alternative referents. This has several implications for root- and rule-based terms. Firstly, two root- and rule-based terms which contain the same roots but have different structure can be used. The term ROOM-TEMPERATURE-ELASTIC_CONSTANT is distinct from the term ROOM-TEMPERATURE-ELASTIC-CONSTANT - the former refers to the elastic constant at room temperature, while the latter refers to a constant relating to room temperature elastic (e.g. elastic material held at room temperature). Because these two terms have distinct meanings and distinct structures, both can be represented if necessary, and both can be generated from natural language. Secondly, the structure of root- and rule-based terms implies a larger semantic structure.

The two terms ROOM-TEMPERATURE-ELASTIC_CONSTANT and ABSOLUTEZERO-ELASTIC_CONSTANT both contain the super-root ELASTIC_CONSTANT and both refer to types of elastic constant, i.e. the elastic constants at room temperature and the elastic constants at absolute zero. However, other terms, such as ELASTIC_CONSTANT-TEMPERATURE_VARIATION also contain ELASTIC_CONSTANT but do not refer to types of elastic constant. Instead, they refer to a type of temperature variation. This can be determined from the structure of root- and rule-based terms, allowing a computer to generate term hierarchies and semantic maps.

The structural and hierarchical nature of root- and rule-based terms also allow for more powerful

[3] In a head-final structure, the head phrase follows its dependents. For example, the noun in a noun phrase follows the adjectives which modify it.

searches and analyses of data. Documents can be indexed not just by the words that occur in them, but by salient concepts they discuss. This allows a document describing ROOM-TEMPERATURE-ELASTIC_CONSTANT to be distinguished from one describing ROOM-TEMPERATURE-ELASTIC-CONSTANT - even though the roots that make up these terms are the same, the two documents are describing different concepts. Thus, a user searching in a collection of documents can specify whether they are hoping to find documents on ROOM-TEMPERATURE-ELASTIC_CONSTANT or on ROOM-TEMPERATURE-ELASTIC-CONSTANT. Furthermore, because ROOM-TEMPERATURE-ELASTIC_CONSTANT is more specific than just ELASTIC_CONSTANT, a user may also be able to search for more general concepts as well. Similarly, given a general term such as ELASTIC_CONSTANT, a system can determine more specific related terms such as ROOM-TEMPERATURE-ELASTIC_CONSTANT and ABSOLUTEZERO-ELASTIC_CONSTANT.

6. Applying Root- and Rule-Based Terminologies

In previous sections we have discussed how we propose to build domain terminologies using a root- and rule-based approach. We have described how an algorithm can convert phrases of natural language into structured terms, how key phrases can be extracted, and how this system can be extended and modularized. In this section, we discuss why the root- and rule-based approach we have proposed facilitates the creation of useful terminologies.

It is non-trivial to measure the correctness of a domain terminology. Standard metrics such as precision (the percentage of the output answers that are desirable) and recall (the percentage of all desired answers that are actually contained in the output) may not accurately assess problems without definite answers. Terminologies are only desirable insofar as they represent sets of useful concepts relating to a domain; different uses may lead to different notions of desirability. A terminology does not represent every possible concept used in a domain – instead, it represents only those concepts that are of appropriate specificity for practical purposes, depending on the use case.

This notion of practical validity does not have an objective definition that can be directly measured. Instead, the use determines the validity of terminology in a particular context. For example, a terminology that is based entirely on taxonomy (a scheme of hierarchical categorization) may be useful for some tasks, but for others, it may be desirable to represent information about the properties of terms using a different scheme of classification. In some cases, for example, it may be useful to know that RED is a type of COLOR, while in others it will instead be useful to know that a BALLOON has the property COLOR with value RED. The Root- and rule-based approach and be adapted to support new ways of classification.

More generally, terminologies need to be available on demand to many types of interaction with users. For example, in addition to end-users, a terminology needs to be easily maintained by administrators. The implementation of a terminology may be very efficient on the user end, obtaining the results of search queries very quickly, but be inefficient on the administration end. This is often a detail of implementation, but can be relevant to the formulation of the terminology model.

The root-and-rule-based model that we have described is designed to meet the needs of a variety of general use-cases and be configurable enough to meet the needs of more specific situations. We have considered four general use-cases in our proposal: document entry, document retrieval, curation, and rule changes. These use cases assume a centralized database containing the core terminology. The relationships between these use cases and the data are shown in Figure 8. The terminology generator in Figure 8 is shown is the same system shown in Figure 1. The desirable components of a terminology system such as this are described in Section 1.2

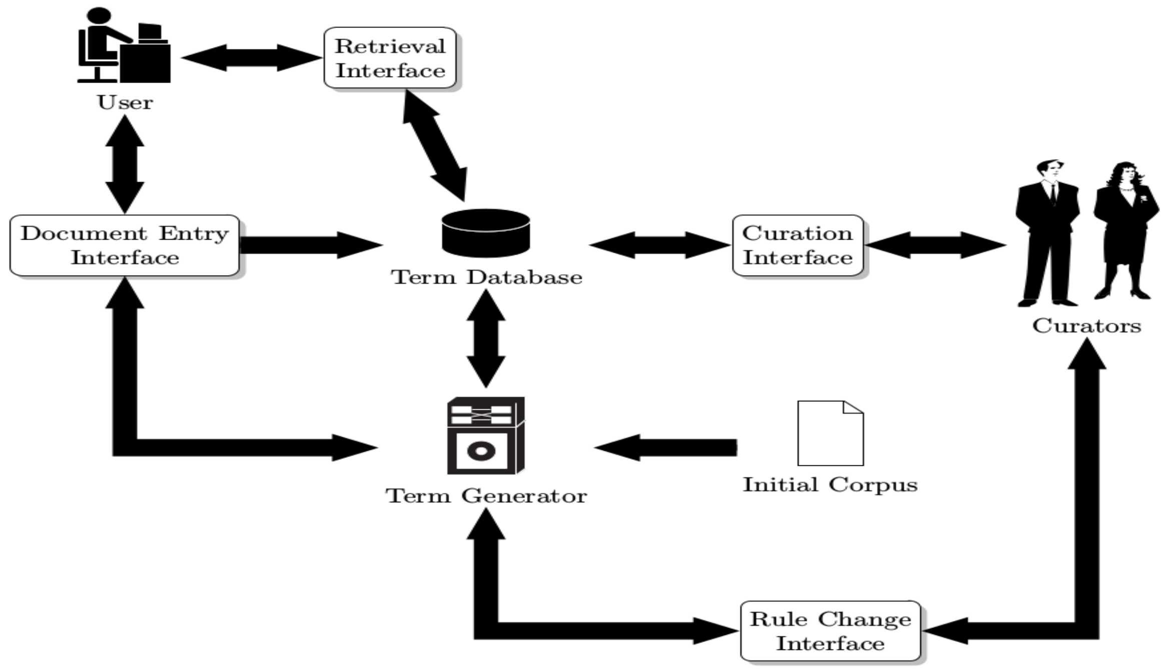

Figure 8. A strategy for use-cases to create and manage root- and rule-based terminologies

6.1 Document Entry Interface

We have proposed that a document entry interface should provide functionality that allows a user to upload a document to the system. The system should then determine the terms in the document that match pre-existing terms, extract any new terms from the document, and allow the user to edit these terms.

We propose that a user should be able to edit the results of document entry by removing those terms that do not apply to the uploaded document, or by adding new terms that the algorithm did not find.

A document entry interface is dependent on the ability to quickly identify terms in a single document. That is, an automated system that allows for single document entry must be robust to corpora containing few documents. The terminology must be able to be built one document at a time, with no dependency on large corpora. Our proposed root-and rule-based system would take advantage of redundant methods of terminology extraction in order to work with different-sized corpora. In addition to extracting terms with a more general-purpose terminology extraction algorithm (see Section 4), our proposal also takes advantage of the structured nature of terms to find terms in new documents that are structurally similar to previous terms. This often allows the root- and rule-based method to identify useful new terms even without statistical evidence.

6.2 Document Retrieval Interface

In our proposed document search and retrieval interface, a user should be able to enter one or more terms and receive a listing of documents containing the given terms. In the simplest case, the user simply inputs a list of terms, and the system locates all of the documents in the database containing the given terms. More complex search systems are also possible, allowing for additional refinement of search criteria.

Structured terms improve the potential of search systems. The compositional nature of terms in this model means that users can make semantic searches, rather than simply searching for the presence of a collection unrelated words. That is, instead of searching for a document that contains the words “red”, “tree”, and “leaf”, a user can easily identify the exact concept in question and search for RED-TREE_LEAF. This allows the user to make much more succinct and semantically rich search queries, producing narrower and more relevant result sets.

The system by which terms are generated from natural language phrases can also be used to improve the usability of user search interfaces. Rather than requiring that users search directly for structured terms in the database, which requires an understanding of the way that root- and rule-based terms are formed, the interface can instead allow users to input phrases of natural language. For example, if the user inputs “the red leaves of a tree”, the interface can quickly generate suggested terms, such as RED-TREE_LEAF, meaning that users only need a passive knowledge of term structure in order to make advanced use of the interface, while still allowing for unambiguous searches.

6.3 Curation Interface

A curation interface allows a team of curators to make changes to the terminology. Curators may either be a select team of administrators, or volunteer end-users. That is, it is possible in some cases to crowd-source curation. Depending on the resources available for a particular system, it may be desirable to have dedicated moderators or to allow end-users to make their own changes to the database. We make no suggestions as to which is more generally preferable, but instead describe a terminology system that is able to handle both.

Curation may be partially dependent on search, as discussed above, in subsection 6.2, as curators need to be able to locate terms to change. However, curators may benefit from more than a document retrieval system, as they should be able to examine the complete structure of the terminology. For example, curators may wish to view taxonomic relationships between terms in order to ensure that the taxonomy is structured correctly. Curators should be able to make changes to the structure, as well as to individual terms in the terminology.

A root- and rule-based terminology interacts well with this sort of curation interface. The semantic nature of terms can be used to determine the taxonomic structure of the terminology implicitly (as discussed in Section 2, modifiers (roots other than the head) add specificity, and unmodified terms are equivalent to hyponyms relative to their modified variants). Additional relationships between terms are represented through compounded terms (semantic clusters represented by terms that have been combined using the delimiter ':'), allowing for graph-based visualizations, such as that shown in Figure 9. A visualization such as this might be derived from the following two terms: ELECTRON-CYCLOTRON-CURRENT_DRIVE:NONINDUCTIVE-CURRENT_DRIVE and ION-CYCLOTRON-CURRENT_DRIVE. By parsing these terms into roots, an algorithm or a user can easily determine that both terms represent types of CURRENT_DRIVEs, and that ELECTRON-CYCLOTRON-CURRENT_DRIVE and ION-CYCLOTRON-CURRENT_DRIVE are types of CYCLOTRON-CURRENT_DRIVE. Furthermore, the user interface can show that there is some relationship between NONINDUCTIVE-CURRENT_DRIVE and ELECTRON-CYCLOTRON-CURRENT_DRIVE. This information is all determined from the structure of these three terms, and the composition of the roots.

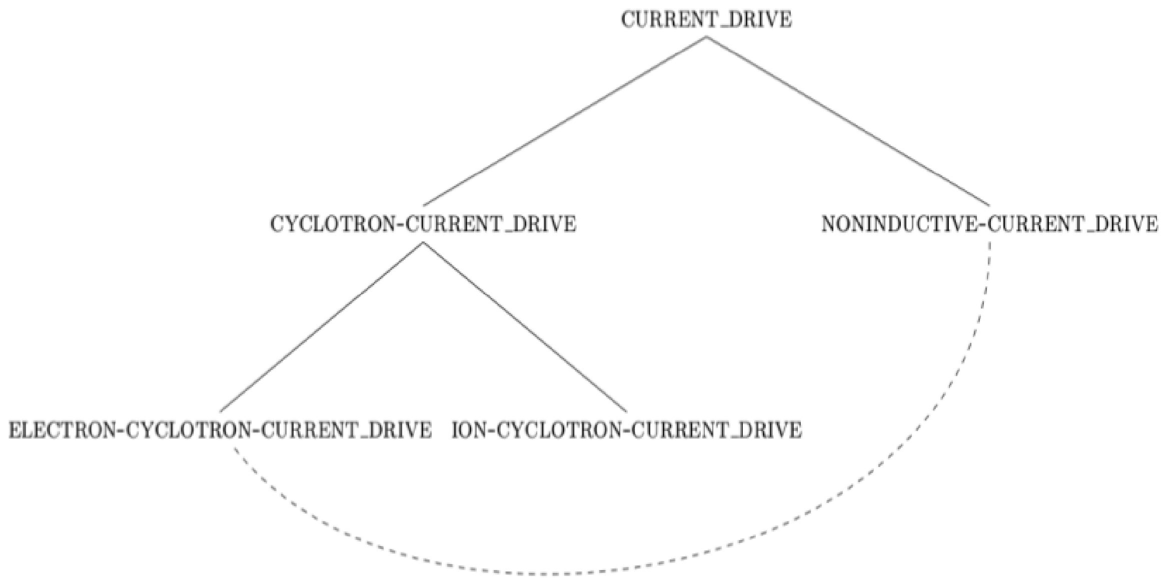
CURRENT_DRIVE
CYCLOTRON-CURRENT_DRIVE
NONINDUCTIVE-CURRENT_DRIVE
ELECTRON-CYCLOTRON-CURRENT_DRIVE
ION-CYCLOTRON-CURRENT_DRIVE

Figure 9. Visualizing Term realtionships

The implicit and explicit relationships between terms in this model allow for changes to terms without requiring manual restructuring of the hierarchical structure. For example, if the system erroneously created the term RED-LEAF_TREE instead of RED-TREE_LEAF, a curator could make this change without needing to manually relocate the term to its proper taxonomic position beneath LEAF. Instead, the curator only needs to change the term RED-LEAF_TREE to RED-TREE_LEAF and the fact that RED-TREE_LEAF is a type of LEAF can be automatically inferred.

6.4 Rule Change Interface

Rule change interfaces are desirable in many evolving situations. Due to the constant changes in knowledge and vocabulary that occur in all domains, it is sometimes necessary to update the way that the rules are used to generate terms. This interface may be particularly useful during the infancy of this technology. In our proposal, it may be desirable to create new delimiters with new meanings, change the behavior of delimiters, or change the way that terms are selected. In addition, it may become feasible to make changes due to improvements in technologies. In our proposal, this process does not require a complete restructuring of the system. Instead, only an interface which allows for the replacement of individual components of the system, as described in Section 5, is necessary.

6.5 Generating Ontologies

As discussed in Section 1, one of the motivations for a root- and rule-based method of terminology generation is practical ontology extraction for the semantic web as well as identifying, tracking, and predicting changing trends. The root- and rule-based terminologies we have discussed are well-adapted for use in ontologies. Though the methods we describe do not immediately output a complete ontology, the structured nature of terms in root- and rule-based terminologies means that a terminology can easily be extended into the skeleton of an ontology. As shown in Figure 9, some of the relationships between terms can be inferred from term structure. These relationships are primarily taxonomic, but compounded terms reveal additional connections that may inform the structure of a complete ontology.

Some of the relationships that structured terms encode are underspecified. For example, the term ELECTRON-CYCLOTRON-CURRENT_DRIVE:NONINDUCTIVE-CURRENT_DRIVE entails that there is a relationship between ELECTRON-CYCLOTRON-CURRENT_DRIVE and NONINDUCTIVE-CURRENT_DRIVE. However, it does not specify what that relationship is. An ontology may need to supply this additional information, but the presence of the relationship is already known. Because of this, a root- and rule-based terminology provides many of the relationships necessary for a domain ontology. A system could generate a simple ontology by adding labels to these relationships and adding any relevant edges that may have been left out in terminology generation.

Ontologies are usually dependent on the terminologies that inform them. By basing ontologies on root- and rule-based terminologies, it is possible to create ontologies that are human- and machine-readable. Because language, including the vocabulary used in most domains, is constantly evolving, ontologies also need to evolve. Because root- and rule-based terminologies are adaptable to new needs and to evolving vocabulary as discussed in Section 5, they are superior to ad hoc terminologies for constructing practical domain ontologies.

It is possible to partially automate the creation of domain ontologies from terminologies. Statistical analyses such as co-word analysis (Kostoff 1993; Coulter et al. 1996; Coulter, Monarch, and Konda 1998) can suggest relationships among the terms in a terminology, though not label these relationships. These methods associate relationships between terms which occur together within a fixed window. For example, if the terms ELECTRON-CYCLOTRON-CURRENT_DRIVE and NONINDUCTIVE-CURRENT_DRIVE occur together more often than is expected based on the frequencies of the individual terms, then it is likely that there is a semantic connection between them. The above analyses do not automatically label these relationships, but crowd-sourced methods can help to assign labels to common relationships and create training sets for future automatic labeling techniques. Though there is currently no fully-implemented application that extends root- and rule-based terminologies into domain ontologies, a system that uses a root- and rule-based approach as well as co-word analysis and crowd-sourced labeling is currently in development. Moreover, the terminologies generated by this approach can be used with current applications, both commercial and freely available, to produce terminological networks much like the ones produced by co-word analysis using tools such as Leximancer and Gephi (Smith and Humphreys, 2006; Mathieu Bastian 2009).

7. Conclusion

Our root- and rule-based approach present several advantages for the development of domain-based terminologies that are not available in semi-structured models, while still maintaining both human- and machine-readability. The primary advantage of root- and rule-based terms is that they allow one to consistently and clearly represent important domain concepts. Root- and rule-based terms are compositional, allowing for the division of terms into their component parts for searching, selecting new terms, or deriving relationships between terms.

Our proposed root- and rule-based model is also highly modular, meaning that different users can easily adapt and maintain terminological systems. Components may be updated with technological advances, the introduction of new uses, or adaptations based on how systems are being used in real-world situations. The model is also designed with many important use-cases in mind, including search, document uploading, and curation. This allows the system to be practical for the needs of different users, for the administration, and for developers and scientists hoping to expand upon a root- and rule-based system.

The model presented here is linguistically motivated, and follows from many aspects of linguistic theory, including syntax, semantics, and pragmatics, allowing it to connect on a fundamental level with the way that humans actually use language, rather than with mathematical constructs transparent only to machines. Like language itself, this model is evolutionary, use-based, and compositional, designed with practical needs rather than purely theoretical constructs in mind.

References

Berners-Lee, Tim, James Hendler, and Ora Lassila. 2001. “The Semantic Web.” *Scientific American*.

Bhat, Talapady N. 2010. “Building Chemical Ontology for Semantic Web Using Substructures Created by Chem-BLAST.” *International Journal on Semantic Web and Information Systems* 6 (3): 22–37.

Bhat, Talapady N., Laura M. Bartolo, Ursula R. Kattner, Carelyn E. Campbell, and John T. Elliott. 2015. “Strategy for Extensible, Evolving Terminology for the Materials Genome Initiative Efforts.” *Journal of Materials*, no. 8: 1866–75.

Clark, Herbert H., and Deanna Wilkes-Gibbs. 1986. “Referring as a Collaborative Process.” *Cognition*, no. 22: 1–39.

Coulter, Neal, Ira Monarch, and Suresh Konda. 1998. “Software Engineering as Seen Through Its Research Literature: A Study in Co-Word Analysis.” *Journal of the Association for Information Science and Technology* 49 (13). New York, NY: John Wiley & Sons, Inc.: 1206–23.

Coulter, Neal, Ira Monarch, Suresh Konda, and Marvin Carr. 1996. “An Evolutionary Perspective of Software Engineering Research Through Co-Word Analysis.” CMU/SEI-95-TR-019. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

Feigenbaum, Lee, Ivan Herman, Tonya Hongsermeier, Eric Neumann, and Susie Stephens. 2007. “The Semantic Web in Action.” *Scientific American*.

Fellbaum, Christiane, ed. 1998. *WordNet: An Electronic Lexical Database*. Cambridge, MA: MIT Press.

Frantzi, Katerina T., Sophia Ananiadou, and Jun-ichi Tsujii. 1998. “The C-Value/NC-Value Method of Automatic Recognition for Multi-Word Terms.” In *Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries*, 585–604. London: Springer-Verlag.

Frege, Gottlob. 1884. *The Foundations of Arithmetic*. 1980th ed. Evanston, IL: Northwestern University Press.

Hall, Johan. 2006. “MaltParser: An Architecture for Labeled Inductive Dependency Parsing.” Master’s thesis, Växjö University.

Hearst, Marti A. 1992. “Automatic Acquisition of Hyponyms from Large Text Corpora.” In *Proceedings of the 14th Conference on Computational Linguistics*, 2:539–45.

Hudson, Richard A. 2004. “Are Determiners Heads?” *Functions of Language* 11 (1): 7–42.

Knechtel, Martin, and Rafael Peñaloza. 2010. “A Generic Approach for Correcting Access Restrictions to a Consequence.” In *7th Extended Semantic Web Conference*. The Semantic Web: Research and Applications.

Koivunen, Marja-Riitta, and Eric Miller. 2001. “W3C Semantic Web Activity.” In *Semantic Web Kick-Off in Finland: Vision, Technologies, Research, and Applications*, 27–44. Helsinki Institute for Information Technology.

Kostoff, Ronald N. 1993. “Co-Word Analysis.” In *Evaluating R&D Impacts: Methods and Practice*, 63–78. Springer.

Liu, Xueqing, Yangqiu Song, Shixia Liu, and Haixun Wang. 2012. “Automatic Taxonomy Construction from Keywords.” In *ACM Conference on Knowledge Discovery and Data Mining (Kdd 2012)*. Beijing, China.

Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. “The Stanford CoreNLP Natural Language Processing Toolkit.” In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 55–60.

Mathieu Bastian, Mathieu Jacomy, Sebastien Heymann. 2009. “Gephi: An Open Source Software for Exploring and Manipulating Networks.” In *International Aaai Conference on Weblogs and Social Media*, 361–62. Association for the Advancement of Artificial Intelligence.

“Modeling Across Scales.” 2015. Warrendale, PA: The Minerals, Metals & Materials Society.

Montague, Richard. 1988. “The Proper Treatment of Quantification in Ordinary English.” In *Philosophy, Language, and Artificial Intelligence*, 2:141–62. Studies in Cognitive Systems. Amsterdam: Springer Netherlands.

Nivre, Joakim. 2003. “An Efficient Algorithm for Projective Dependency Parsing.” In *Proceedings of the 8th International Workshop on Parsing Technologies (Iwpt 03)*, 149–60. Nancy, France.

Overton Jr, W.C., and J. Gaffney. 1955. “Temperature Variation of the Elastic Constants of Cubic Elements.” *Phys. Rev.* 98: 969–77.

Park, Youngja, Roy J. Byrd, and Branimir K Boguraev. 2002. “Automatic Glossary Extraction: Beyond Terminology Identification.” In *Proceedings of the 19th International Conference on*

Computational Linguistics (Coling '02), 1:1–7.

Plant, Anne L., John T. Elliott, and Talapady N. Bhat. 2011. "New Concepts for Building Vocabulary for Cell Image Ontologies." *BMC Bioinformatics* 12 (487).

Polguère, Alain, and Igor Mel'čuk, eds. 2009. *Dependency in Linguistic Description*. Vol. 3. Studies in Language Companion Series. John Benjamins.

Proux, Denys, Francois Rechenmann, Laurent Julliard, Violaine Pillet, and Bernard Jacq. 1998. "Detecting Gene Symbols and Names in Biological Texts: A First Step Toward Pertinent Information Extraction." *Genome Informatics* 9 (72–80).

Rindflesch, Thomas C., Lawrence Hunter, and Alan R. Aronson. 1999. "Mining Molecular Binding Terminology from Biomedical Text." In *Proceedings of the Amia 1999 Symposium*, 127–31.

Shannon, Claude E. 1948. "A Mathematical Theory of Communication." *Bell System Technical Journal* 27 (3): 379–423.

Sharma, Deepik. 2010. "Stemming Algorithms: A Comparative Study and Their Analysis." *International Journal of Applied Information Systems* 4 (3). New York: Foundation of Computer Science: 7–12.

Smith, Andrew E., and Michael S. Humphreys. 2006. "Evaluation of Unsupervised Semantic Mapping of Natural Language with Leximancer Concept Mapping." *Behavior Research Methods* 38 (2): 262–79.

Swartz, Aaron. 2013. *Aaron Swartz's a Programmable Web: An Unfinished Work*. Morgan & Claypool.

Tesnière, Lucien. 1959. *Éléments de Syntaxe Structurale*. Paris: Klincksieck.

Witschel, Hans Friedrich. 2005. "Terminology Extraction and Automatic Indexing - Comparison and Qualitative Evaluation of Methods." In *Proceedings of Terminology and Knowledge (Tke)*, 1–12.

Wu, Ho Chung, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok. 2008. "Interpreting TF-IDF Weights as Making Relevance Decisions." *ACM Transactions on Information Systems (TOIS)* 26 (3): 1–37.

Yoshida, M., Y. Sakamoto, H. Takenaga, S. Ide, N. Oyama, T. Kobayashi, and Y. Kamada. n.d. "Rotation Drive and Momentum Transport with Electron Cyclotron Heating in Tokamak Plasmas." *Phys. Rev. Lett.* 103 (6). American Physical Society.

Appendix A: Rules

The enumerated rules listed below were originally published in (Bhat et al. 2015).

1. Forming roots:

    1. Use all roots in singular form except where plural form is used more frequently.
    2. Avoid using special characters (such as ’ : _ - = / \) as a part of a root.
    3. Avoid the use of modifiers as roots
    4. Use abbreviations only when they are widely accepted across many related disciplines and when they are unambiguous in their meaning. See Rule [itm:ambiguities] for exceptions when acronyms are embedded in a super root. Use uppercase for all acronyms except for atomic symbols.
    5. For similar expressions choose a shorter equivalent as a root.

2. Forming super roots:
   A super root is formed when the roots involved do not have a preferred discriminating power and semantics to serve as node names of a data-graph or as RDF elements except in special circumstances.
    1. Super roots are concatenated by an underscore to indicate its compound semantics and its ability to be parsed into individual roots only under unusual conditions. If a super root is comprised of roots that are not specific when considered individually, then refer to tethered roots (see Rule [itm:tethered]).
    2. When a root of a super root functions like a hierarchical classifier to another root then also include the classified root in the super root so that automated parsers can recognize the hierarchy. To order roots within a super root, unless there already exists a well-accepted alternate convention, use rule [itm:ordering].
3. Forming tethered roots:
    a) Create tethered roots when a root is a qualifier of another root and the semantics of any root on its own may not be of interest in a database or data repository search. Tethered roots are formed to indicate that the roots involved need be considered collectively, rather than individually, in order to derive their semantics. For this reason, roots in a tethered root are written contiguously to avoid inadvertent separation by automated methods. Since tethered roots are comprised of qualifier and qualified roots, following a general convention of root-based construction of English language words, we use their intrinsic qualifier-qualified relationship to order their roots.
    b) A root may appear in more than one tethered root.

   Tethered roots may also provide a way to avoid the use of stop words in a compounded root. That is, move the word from the right of a stop word to the left, drop the stop word, and place the qualifier before the qualified.
4. Forming terms from roots: Terms are formed by concatenating two or more roots, super roots or tethered roots using a hyphen (-) so that automated methods may re-generate their roots when necessary. We suggest to order roots of a term by classifier-classified relationships (See Rule 6) which is also a general convention in English, as in police dog or technical paper unless there is a different well accepted convention.
5. Avoiding ambiguities and redundancies
    a) Avoid using ambiguous acronyms. Instead clarify their meaning by qualifying them

with a classifier 'root' to form a super root or a tethered root or use the complete phrase.

b) Avoid the inclusion of redundant words in a term.

6. Ordering roots in a term – classifier-classified rule: Roots (super root, tethered root and root) within a term are organized by a left to right, semantic top-down, classifier-classified hierarchy. In general, classifier and classified roots are expected to have one-to-many relationships where, in a rules-based approach, for example, the root alloy is a classifier for many materials. Rule 16 deals with instances where a relationship is not obvious or when a relationship changes over time due to the addition of new terms. In short, a hierarchy is not absolute but rather, it varies with the number of relevant use-cases.

   a) One way to identify classifier and classified roots in a term is to arrange the terms with an embedded hierarchical top-down, level-based classifier (for each 'classifier' term there exists several possibilities of 'classified' terms) statement with a hyphen between classifier and classified terms (e.g., MODELINGSOFTWARE-VASP, MODELINGSOFTWARE-ABINIT). On sorting these terms, classified roots appear as the fast varying strings (VASP, ABINIT) and their classifier roots appear as the slow varying term (MODELINGSOFTWARE). Automated methods may use this feature to develop hierarchical data models that can be presented as data-graphs or RDF or used for auto-complete to select terms for reliable search results.

   b) When a classifier-classified relationship does not exist among the roots, place them in an alphabetical order.

7. Creating roots and terms with similar, multiple, or complex meanings: Following rule 1, use a shorter root for words with similar meaning whenever possible. A root embedded in a term can help automated methods, such as co-word analysis, natural language processing, and text-mining, to identify related semantic classes. To facilitate this process, it is recommended: a) to limit the use of synonymous roots; b) if necessary, clarify the semantics of a root by appending it with a classifier-root.

8. Reusing terms to create compounded terms: Create terms by combining roots so that terms have clear semantics. Avoid terms that are broad and general in meaning. Create terms that can serve as 'semantic expressions' in use-cases. A rule of thumb is to attempt to form terms with three roots and, if needed, combine between two and five terms to form suitable semantic expressions.

9. Creating compounded terms that identify a group of objects in the terminology: Compound terms serve as 'use-cases' defining semantic expressions of terms and they are formed by concatenating two or more terms using a colon (:) as a special delimiting character. Compounded terms that are overly specific are unlikely to be reused. It is advised to limit the number of terms in a compounded term to between two and five terms. Compounded terms may point to persistent identifiers (PIDs), such as DOIs (Digital Object Identifier) for query purposes. Compounded terms may be used by database providers or repository administrators to cluster, identify, and display related items using messages like 'related to items that you have viewed'.

    a) Use classifier-classified hierarchical Rule 6 to decide the order of terms in a compounded term.
    b) When creating compounded terms, give importance to ‘use case-on-demand’ hierarchies, which are case-based rather than fixed schema-based hierarchies. Order a term so that a term to the left has one-to-many relationship with the term to its right.

10. Providing the reference of any paper that supports the use of the new term(s) you are creating. The reference may serve as a ‘definition’ of the term as well may demonstrate use of the term within a context.
11. Design for readability of compounded terms: Use uppercase for the first letter of a term and use lowercase for all the rest unless a root is a short form or a symbol.
12. Provide usage statistics for terms: For each term in a database or repository, store its usage statistics for users to inspect, along with the terms. These frequencies may allow a user to avoid terms that are used infrequently.
13. Provide semantic context of terms and compounded terms: In the database, also keep and display a bibliographic reference and/or DOI to illustrate the use and semantics of the term. This reference may also be used as the basis to build use-case-specific compounded terms or segments of data-graphs.
14. Identify new terms introduced by users as well as flag terms if no documentation is provided. (See Rule 10)
15. Allow the creation of dialects: Terms that do not follow the rules may also be created as local dialects when necessary. Dialects may facilitate a gradual evolution of rule-based terminology and the rules in a crowd-sourced environment.
16. Curate and validate terminology and compounded terms on a regular basis: Dialects are important components of the proposed method for terminology building. Therefore, accepting or removing dialects as terminology must be facilitated by public resource providers who act as caretakers. Redefining super roots, tethered roots and classifier-classified relationships among roots are all important steps of the evolution process of the proposed term building effort. Database developers and repository administrators need to have an established mechanism for regular updates to support a smooth evolution process. Frequency of usage and the semantic context of terms are useful factors to monitor in such an evolution process.
17. Apply new technologies that have been adopted widely: Explore whether new data technologies may require the rules to be updated.

Appendix B: Noun Phrase Syntax and Semantics

One of the key components of any automated terminology generation system is to find and represent important concepts. Typically, the source of information that leads to these representations is a series of natural language texts, such as a corpus of scientific articles. For our root- and rule-based approach, representing these natural language descriptions is dependent on an effective model of semantics, which we discuss in this chapter. A complete model of natural language semantics (i.e. a perfect representation of meaning) is most likely beyond the scope of contemporary linguistics and computer science, so we will focus on a limited subset of semantics – namely, the compositional semantics of noun phrases. For the purposes of this paper, we consider a noun phrase to consist of a noun in addition to all of its modifiers and any determiners (such as *an*, *the*, *this*, and numbers). For example, *the green vase* is a noun phrase consisting of an article (*the*), a modifier (the adjective *green*), and a noun (*vase*). Some of the theory discussed in this section will apply to verb phrases and other syntactic categories, but noun phrases are ideal in that they are relatively concise and in that they are often clear representations of key concepts and commonly appear as headwords or phrases in technical glossaries.

A very simple model of noun phrase semantics would be to simply represent words as themselves or their lemma-forms. For example, the words *leaf* and *leaves* could both be represented as LEAF. For more complex noun phrases, such as *green leaf*, this method becomes more problematic. We could create a representation such as GREENLEAF, a term used specifically for green leaves, but this model does not unambiguously reveal that the meaning of GREENLEAF is related to the meanings of GREEN and LEAF. For this reason, we also model the syntax of noun phrases. Syntax can be represented in several ways, but two of the most common methods are phrase structure grammar and dependency grammar. These models show the relationships between words in a phrase, revealing for example that the word *green* in the phrase *green leaf* is an adjective modifying the noun *leaf*. Syntax and compositional semantics are related concepts; the interpretation of a phrase is derived partially from its syntactic structure (i.e. the organization of the words in a phrase or sentence) (Montague 1988), and so using syntax as a proxy for semantics is reasonable.

Though syntax may relate to semantics, it is not a perfect stand-in. There is not a one-to-one relationship between syntax and semantics – multiple syntactic forms can have the same semantic meaning, and the same syntactic form can have multiple meanings. Because of this, deriving meaning purely from the way that words are put together can lead to ambiguities. Consider the noun phrases in examples 1 through 4; though these sentences have different syntactic structures and are composed of different words, they have very similar meanings.

1) the tree’s red leaves
2) the tree’s leaves that are red
3) the leaves of the tree that are red
4) the red leaves of the tree

If we represent all of these phrases as the unordered set of “important” words in each sentence, all of these sentences could be represented as {*tree*, *red*, *leaves*}. However, unwanted phrases will also be represented this way; for example, *the red tree’s leaves* will also be represented as {*tree*, *red*, *leaves*}. Clearly some syntactic information is necessary in order to create a sufficiently discriminating system. A simple syntactic model is too discriminating; the words in these four sentences are ordered differently, are contained within different syntactic constituents, and are different in grammatical form (*tree* occurs both in its base form and in its possessive form *tree’s*). Thus, we need to develop a model of syntax and semantics that normalizes these differences while still separating them from other phrases. We will begin by discussing how common models of syntax, such as dependency grammar, can be used as a basis for generating structured terms.

B.1 Dependency Grammar

Modern Dependency Grammar (DG) was first described in (Tesnière 1959), and has since become one of the primary syntactic models used in computational linguistics. The key concept in DG is, of course, **dependency**, which is a one-to-one correspondence between morphemes in which every morpheme is **headed** by some other morpheme. For the purposes of this paper, we define a morpheme as the smallest unit of language that has meaning, including simple unbound words such as *leaf* as well as affixes such as the *-s* in *trees*. The **head** of a phrase carries that phrase's syntactic category (i.e. the phrase *the green vase* behaves as a noun and is considered a **noun phrase** because it is headed by *vase*, which is a noun on its own). In DG, the head of a sentence is the verb, and all other morphemes are either direct or indirect dependencies of the main verb. The **dependencies** of a morpheme are those headed by it. For example, in the phrase *the green vase*, *vase* is the head of *green* and *the*, so the dependencies of *vase* are *green* and *the*. If *green* had its own dependencies, these would be indirect dependencies of *vase*.

DG can be represented graphically as a tree structure with the verb as the parent node and dependencies represented as daughter nodes. However, since we are dealing exclusively with noun phrases, we will use a noun as the parent and adjectives, determiners, relative clauses, and other nouns as dependencies[4].

The four noun phrases from examples (1) through (2) are represented as dependency trees in Figures 10 through 13. *Leaves* plays the role of the head because it is the primary (most general) concept represented by the phrase

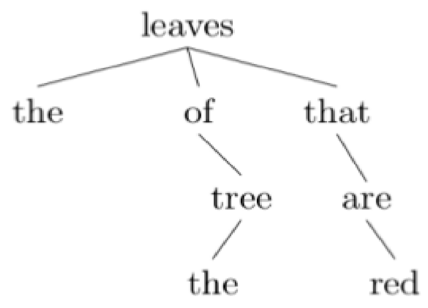

Figure 10: Dependency representation of "the leaves of the tree that are red"

These trees represent both syntactic dependency and linear word order. Each node in the tree represents a word; the dependencies of that word are all daughter nodes. The root, at the top of the tree, is the head and is not a dependency of any other words within the context of these phrasal trees (namely, it does not qualify any other words). The word order can be recovered through an in-order traversal (begin with the left-most node and continue to the right).

[4] According to Hudson (2004), among others, phrases such as "the tree's leaves" and "the green vase" are actually determiner phrases and not noun phrases, and are headed by determiners such as "the" or "a". Though there is significant evidence for this, and most modern generative syntacticians prefer this analysis, we continue to use the noun phrase analysis throughout this paper due to its more intuitive simplicity and for its continued prevalence in computational linguistics. Furthermore, because we will end up ignoring determiners (see Appendix A.3), the distinction between determiner phrase and noun phrase analyses does not have an effect on the model we describe here.

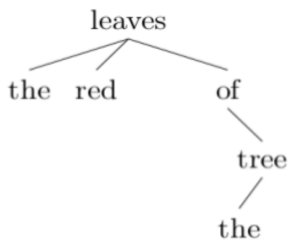

Figure 11: Dependency representation of "the read leaves of the tree"

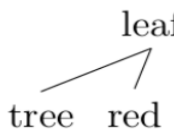

Figure 12: Nomralized dependency representation of "the tree's red leaf"

As noted above, the syntactic and lexical information depicted by these dependency trees is not sufficient to show that these four noun phrases are similar enough to represent the same concept. Structurally speaking, these trees are very different; only the root node (*leaves*) has the same position in every tree, and the remaining nodes are positioned almost everywhere in the tree. This suggests that we cannot use DG on its own to show how closely these sentences are related. However, in the following two sections we will show how we can adapt DG to construct a model of semantics that can be used to normalize noun phrases.

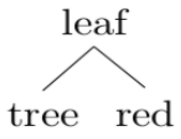

Figure 13: Normalized dependency representation of "the leaves of the tree that are red"

B.2 The Semantics of Dependency

Since we are trying to build a model of semantics, it is necessary for us to delve deeper into syntactic dependency in order to uncover the underlying semantics of noun phrases.

Polguère and Mel'čuk (2009) describe syntactic and semantic dependency as separate, but related concepts. They describes syntactic dependency as the bridge between the semantic form of a sentence (its meaning) and its morphological form (the linear string representation of morphemes that is spoken or written). That is, though syntactic dependencies have additional structure and complexity not apparent in semantic dependencies, the two structures are related. We may, in fact, be able to use the syntactic structure of a phrase to estimate its semantic structure, with additional understanding of the relationship between these two theoretical entities.

Semantics can be described (to a certain extent) using predicate logic (Montague 1988). In terms of dependencies, a predicate's dependents are its arguments. These arguments may themselves be predicates with additional dependents, allowing for recursive structures and complex sentences (Polguère and Mel’čuk 2009). This is an imperfect model for our purposes because it does not precisely correspond to syntactic dependency (which is, computationally speaking, easier to derive) and because it divides all morphemes into relationships (predicates) and entities (arguments). In semantics, it is unclear whether words such as *a* should be treated as predicates, arguments, or something else – in some cases, for example, they are treated as quantifiers (cf. (Montague 1988)). However, (Polguère and Mel’čuk 2009) state that all words in a particular phrase must be connected in a semantic dependency structure, which allows for a more consistent representation.

However, despite the disparity between syntactic and semantic dependency described above, there are some commonalities that are important to the development of the model described in this paper. One of the functions of a predicate is to add specificity to its argument(s). In this sense, we can finally observe a clear similarity between syntactic dependencies and semantics: the dependencies of a morpheme almost always add specificity to the meaning of that morpheme (*green leaf* is more specific than just *leaf*).

Consider Figure 11. The phrase represented by this structure has the meaning “leaves” on a very general level. On a more specific level, it is clear that the leaves in question are possessed objects and that they are red. The possession relationship is made more specific by the dependencies of the *’s* morpheme – the tree is the possessor of the leaf, and the tree is made more specific through the article *the*, which shows that the tree in question is a specific tree in the common ground between the speaker and the listener (or between the writer and reader). We will refer to this as a semantic specification relationship, because the parent node is made more specific through its daughters.

Note that unlike semantic dependency as described in (Polguère and Mel’čuk 2009), the specificity relationship always travels downward in the tree, and perfectly matches syntactic dependency. Semantic specificity is an ideal semantic model for our system, despite the fact that we still cannot directly explain the similarities between the four noun phrases in examples 1 through 4, since the trees in Figures 10 and 11 are still different. In order to explain these similarities, we need to normalize our representations.

B.3 Normalizing Similar Structures

The primary differences between the four example phrases we have been discussing so far are found in the presence or absence of **function morphemes** – that is, grammatical units (such as the possessive morpheme *’s*) that do not correspond to any real-world meaning, but instead serve primarily to indicate grammatical relationships. Though these morphemes do affect the semantics of the overall phrase, they are primarily used to allow for different word-orders and slightly nuanced meanings. However, we can recover most of the meaning of a noun phrase without any of the function morphemes.

We create the four “collapsed” trees in Figure 14 through Figure 17 by removing all of the function morphemes, and leaving only content morphemes (grammatical units that do correspond to real-world meaning, i.e. some particular concept, action, or trait).

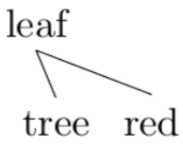

Figure 14: Collapsed representation of the tree's red leaves

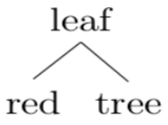

Figure 15: Collapsed representation of the tree's leaves that are red

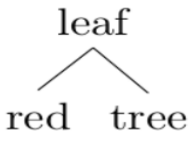

Figure 16: Collapsed representation of the leaves of the tree that are red

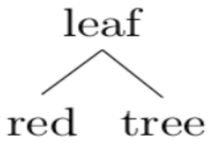

Figure 17: Collapsed representation of the red leaves of the tree

At this point, the similarities between these four structures are quite clear: each one has *leaf* (the head of the noun phrase) as the root, and two dependencies: the words *red* and *tree*, albeit in different positions. However, in our semantic model, linear order does not impact the meaning of the noun phrase as a whole (only vertical order does). Thus, all four of these structures produce the same meaning, namely that of a leaf specified by both *tree* and *red*. Note that there is a trade-off to removing function words: we do not know what type of relationship there is between *leaf* and *tree*, only that a relationship exists, or that it is a generic relationship such as type-of (e.g. a tree-type-of-leaf). Relationships such as synonymy (words that are synonyms), meronymy (words that represent a part of another concept), and possession are not detected. However, we accept this trade-off for our purposes: we are looking for important concepts, not specific referents.

B.4 Representing the Model

The model of semantics described above can be represented graphically using tree structures such as those above, in Figure 11 through Figure 14. To do this in such a way that semantically similar phrases are all represented the same way, we need a consistent way to order daughter nodes. The method that we choose is largely irrelevant, so long as the order of daughter nodes is independent of the phrase's original word order. A trivial method is to order the nodes alphabetically; this is what we will do for now, meaning that sentences 1 through 4 can all be represented using Figure 18.

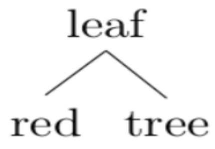

Figure 18: Universal representation for sentences 1 through 4)

However, this representation is not ideal. From a computational standpoint, it is acceptable: it represents unambiguously all of the information we need to know about a noun phrase, including the semantic information that we want for generating terminologies. These structures can be generated quickly and accurately using dependency parsers (Nivre 2003). However, they are *not* easily read by humans without linguistic training. Reading a dependency tree requires an understanding of what a dependency is, as well as how dependency trees are structured. For domain taxonomies, data structures that are not human readable are undesirable. Terminologies need to be read by humans as well as by machines, adding an additional level of challenge to the problem of generating domain terminologies.

We propose to solve this problem by building **structured compound nouns** in such a way that they represent syntactic dependency unambiguously while remaining human-readable. Linguistically speaking, a compound noun is a noun composed of two or more other nouns, such as *dog house* or *airplane*. A **structured compound noun** is a compound noun formed through the application of regular, systematic rules. In English, the way that two compound nouns are formed is not completely predictable. Though *dog house* and *bird house* refer to houses for dogs and birds, respectively, a *fire house* is not a house for fire in the same sense. Other languages, however, have more productive compounding, meaning the same way of creating a compound will have the same meaning in all cases. In Sanskrit and German, for example, compound nouns often (but not always) have predictable meanings based off of their components and how they are combined. In other words, there is a set of rules that determines the meaning of a compound. We can adapt this idea to our semantic model in order to create an easy to read representation that follows from patterns in natural language. Because structured compound nouns are structured representations of phrasal semantics, they can be used to represent terms in a terminology. This produces a terminology of structured terms with predictable meanings based on roots and a set of rules used to combine them.

Our root- and rule-based approach does not use the same rules or patterns as Sanskrit, German, or other languages with agglutinative noun compounds, as the rules in these languages still have many of the issues associated with natural language more generally, including ambiguities and inconsistencies. However, the processes of compounding and composition play a major role in root- and rule-based terminologies.

Bios

T. N. Bhat is a project leader at NIST and one of his recent goals is to develop tools and techniques to enable archiving, searching and sharing scientific information.

Jacob Collard is a PhD student in computational linguistics at Cornell University, where he specializes in interpretable computational models of natural language syntax and semantics and the application of formal methods to natural language processing. Since 2014 he has also been working with the National Institute of Standards and Technology on information retrieval using formal linguistic methods together with conventional natural language processing tools.

Eswaran Subrahmanian is a Research Professor at the Engineering Research Accelerator and Engineering and Public Policy at Carnegie Mellon University (CMU) and a Guest researcher at the Software and Systems Division at National Institute of Standards and Technology. He is a member of the Design Society and the Washington Academy of Sciences, a Distinguished Scientist of the ACM and Fellow of AAAS.

John T Elliott is the group leader of Cell Systems Science Group at NIST. He is currently developing quantitative microscopy techniques for measuring cellular response in a variety of applications.

Ursula R Kattner is a project leader at the Thermodynamics and Kinetics Group at NIST. Some of her current research interests are computational thermodynamics, alloy phase diagram evaluations, metal-hydrogen system, solder systems, superalloy systems.

Carelyn E. Campbell is the group leader of Thermodynamics and Kinetics Group at NIST. Some of the projects she is currently leading are development of informatic tools and repositories for phase-based materials property data, including thermodynamics, diffusion, molar volume, and elastic properties.

Ram D. Sriram is currently the chief of the Software and Systems Division, Information Technology Laboratory, at the National Institute of Standards and Technology. Before joining the Software and Systems Division at NIST, he was on the engineering faculty (1986-1994) at the Massachusetts Institute of Technology (MIT) and was instrumental in setting up the Intelligent Engineering Systems Laboratory.

Ira Monarch has investigated information design and process issues in large-scale engineering programs, both military and industrial, for over thirty years. At the Software Engineering Institute (SEI), he led and participated in projects that developed and used text analytic tools for uncovering patterns and 'identifying risks and failure conditions in software design, architecture, development and maintenance.