Integrating Syntactic and Semantic Tools in Sfy

A Case Study in Lexical Acquisition

Paul M. Heider
Graduate Student
Linguistics Department
SUNY at Buffalo
609 Baldy Hall
Buffalo, NY 14260
pmheider@buffalo.edu

ABSTRACT
Despite the large lexicon of even broad coverage Natural Language Processing systems, there are many missing lexical items. We describe two methodologically distinct approaches to augment the lexicon: the look-up approach and the shotgun approach. Both approaches are framed within Sfy, a new research program to pull together several broad coverage systems.

The classic look-up approach requires compatible electronic sources (e.g. WordNet). Through system hooks into the source, we can pull as much relational and semantic information as Sfy requires. Since WordNet does not overtly provide Sfy with sufficient syntactic role and semantic information, we use the synonyms of an unknown target word that WordNet provides us with to create templates for a new entry in our lexicon.

In a fully realistic model, we cannot always rely on the look-up approach to solve our lexical issues. We need another back-off method. Much like a person intuiting the part-of-speech of a new term, an unknown word presents a syntactic hole to our parser. Only certain parts of speech will fill that hole. We can try to validate all the possible fillers by naively testing every part of speech. The subset of all these sentences that can be parsed informs us of exactly which parts of speech our unknown word can be. Interestingly, the shotgun approach also provides a means to solve the related problem of a familiar orthographic word functioning in an unfamiliar part of speech.

Categories and Subject Descriptors
J.5 [Computer Applications]: Arts & Humanities—Linguistics; I.2.0 [Artificial Intelligence]: General; I.2.4 [Artificial Intelligence]: Knowledge Representation Formalisms & Methods; I.2.6 [Artificial Intelligence]: Learning; I.2.7 [Artificial Intelligence]: Natural Language Processing; I.2.31 [Artificial Intelligence]: Distributed Artificial Intelligence

Keywords
English Resource Grammar, Lexical Knowledge Builder, Minimal Recursion Semantics, Natural Language Processing, Sfy, SNePS

1. INTRODUCTION
In Artificial Intelligence, one of the classic splits is between breadth and depth approaches. Does the algorithm work generally well for general purposes or is it very thorough but only in a small domain? In Natural Language Processing, this dichotomy rears itself as a battle between broad coverage and narrow coverage parsers, lexicons, etc. We have initiated a research program to pull together several theoretically and procedurally similar broad coverage systems into a tool called Sfy. Specifically, the SNePS and the Linguistic Knowledge Building (LKB) systems\(^1\) create the language-independent backbone of my project. We then use the English Resource Grammar (ERG), WordNet\(^2\), and other similar electronic sources to supply the language-specific data.

Despite the broad coverage of ERG, there are always many missing lexical items. Realistically, a complete hand-coded description of English would quickly become dated with the introduction of new terms and adaptation of old terms that is so common to language. Unfortunately for us, the parsing algorithm will balk at any attempt to parse a sentence containing a word unknown to ERG. We need a technique to cope with the unknown. Increasing the number and variety of sources Sfy can access merely extends the distance to the horizon of intractable words. In order to ensure a solution more robust than such a look-up approach, we propose a second, methodologically distinct way to augment the SNePS’s lexicon: the shotgun approach\(^2\). The former approach focuses on using templates from static information retrieval techniques while the latter relies on dynamic contextual clues.

We will first provide an overview of SNePS, LKB, and ERG,\(^1\)See below for a description of all pre-existing systems used in Sfy.\(^2\)Firing a shotgun produces a volley of small shot. The pellets are particularly effective in close quarters but lack both force and precision.
as they stand. The focus will primarily be on their respective interactions with the lexicon. Second, we will discuss the ways in which all three must be modified towards the creation of Sfy. The third section will cover the two lexical acquisition methods. Finally, we will use examples to show the progress made and progress yet to be achieved towards an adaptable lexicon for broad coverage systems.

2. THE BROAD-COVERAGE SYSTEMS

2.1 SNePS

SNePS is a LISP-based system for knowledge-representation, reasoning, and acting. It can be used as a frame system, a logical rule-based system, or a propositional semantic network. In contrast with systems that assign meaning only to the nodes, SNePS has both labeled nodes and paths between them. Inferences can be made based on matching path structures, matching nodes, or some combination of the two. The paths used for this system are uni-directional but invertible. In other words, path can be traversed in either direction but the directions are not necessarily treated equally. The frames discussed below refer to explicitly defined labeled arc sets with an associated semantics. For instance, the proposition “ideas are colorless” is depicted in figure (1). This node structure should be interpreted with respect to the object-property frame defined in figure (2) [8].

2.2 LKB

The LKB is a general-purpose constraint-based parser intended to be used as a grammar and lexicon development environment [2]. It is (mostly) syntactic framework- and language-neutral. In other words, a user provides the system with a language specific grammar and sentences to parse. The LKB then maps the language input to a syntactic and semantic structure that is licensed by the grammar. Most often, these grammars are built according to the framework Head-Driven Phrase Structure Grammar (HPSG, [5]). The basic components of any grammar are as follows:

- **The type system:** Each object in our grammar must be of a specific type. These types fit into an inheritance system to help encode generalizations into the system. Below, we refer to these as POS types.
- **The start structure(s):** Only certain objects are licensed to be at the top of any complete syntactic structure.
- **Lexical entries:** These objects define the relationships between orthographic forms (i.e., characters on a page) and linguistic forms (i.e., a word sense or a formal linguistic description). The linguistic form includes features such as gender and number.
- **Grammar rules:** Given our complete type system, we need restrictions on how our lexical entries can combine to create larger structures. These growing parts will need to eventually resolve into a start structure for an input sentence to be acceptable.

2.3 ERG

ERG is a broad coverage description of English. It encodes semantic information in a flat representation known as Minimal Recursion Semantics (MRS, [3]).

To simplify the analysis, we will recast ERG’s coverage in terms of traditional grammar. For instance, in the type system described above, there are 159 types that would fall under the part of speech (POS) noun. Likewise, there are 183 types we would classify as verbs and 117 that would normally be categorized as adjective (Adj) or adverb (Adv). We have listed some specific types within these traditional parts of speech in table (1). The types shown account for more than half of all types within their respective POS. These are also the elements that must properly combine according to the specifications of the grammar rules (above). For instance, something of the type n−p−n−le (which represents a type of proper noun) would not combine with a determiner. Each lexical entry is first described with respect to its type (see figure (3) for an example).

The lexical entries additionally contain semantic predicates. The predicates are also coded in such a manner as to partially align with our notions of traditional grammar (see table (2)). Importantly, ERG is not concerned with semantics.

The main goals of MRS development have been expressive adequacy, grammatical compatibility, underspecifiability, and computational tractability. In other words, ERG is a description of English that can (1) correctly express the semantics of English (2) in terms that cleanly interact with grammatical information (3) while allowing some aspects to be left ambiguous. Finally, this package must be (4) detailed in a manner that can be verified and analyzed by a computer.

Simply speaking, we can think of a predicate as an n-place relation. isElephant(x) is a predicate used to describe all x that are elephants. isSisterOf(x, y) could be used to describe all x and y such that x is a sister of y.
Table 1: Most Frequent Syntactic Roles by POS

<table>
<thead>
<tr>
<th>Role</th>
<th>Count</th>
<th>% of Its Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noun Role</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n-noun</td>
<td>4220</td>
<td>30.29</td>
</tr>
<tr>
<td>n-constant</td>
<td>4005</td>
<td>28.74</td>
</tr>
<tr>
<td>Verb Role</td>
<td></td>
<td></td>
</tr>
<tr>
<td>v-noun</td>
<td>1555</td>
<td>23.06</td>
</tr>
<tr>
<td>v-constant</td>
<td>791</td>
<td>11.73</td>
</tr>
<tr>
<td>adj/adv Role</td>
<td></td>
<td></td>
</tr>
<tr>
<td>adj-constant</td>
<td>2863</td>
<td>49.84</td>
</tr>
<tr>
<td>av-advvp</td>
<td>685</td>
<td>11.93</td>
</tr>
</tbody>
</table>

Table 2: Most Frequent POS for Predicates

<table>
<thead>
<tr>
<th>POS</th>
<th>Count</th>
<th>% of All Preds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noun</td>
<td>7827</td>
<td>42.99</td>
</tr>
<tr>
<td>Verb</td>
<td>5481</td>
<td>30.10</td>
</tr>
<tr>
<td>Adj/Adv</td>
<td>4291</td>
<td>23.57</td>
</tr>
</tbody>
</table>

at the word level. Rather, ERG (and even the LKB) are concerned with compositional semantics, or the semantics of non-elementary elements. Thus, neither ERG nor the LKB provide information as to the meaning of any particular predicate. They are focused on explaining what influence particular predicates have on the overall meaning of a sentence.

Finally, the predicates used in ERG have argument structures\(^5\) tied to them. Because a single predicate can have multiple argument structures associated with it, the numbers in table (3) are less precise. In the table, we can see the number of arguments attributed to predicates of a given POS. In other words, of the 6,671 predicates with a single argument, 99.13% of them are nouns. Of only noun predicates, 85.75% have a single argument. This makes noun the predominant POS for predicates with a single argument. The predominant POS for each argument structure is in bold.

2.4 A Sketch of the Lexicon

\(^5\)Much like computer functions are associated with a list of arguments, certain parts of speech also require arguments. A description of the number and type of these associated words is called the argument structure.

Table 3: Predicate Arguments and Distribution

<table>
<thead>
<tr>
<th>Arguments</th>
<th>ARG0</th>
<th>ARG1</th>
<th>ARG2</th>
<th>ARG3</th>
<th>ARG0</th>
<th>ARG1</th>
<th>ARG2</th>
<th>ARG3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARG0 ARG1 ARG2 ARG3</td>
<td>ARG0</td>
<td>ARG1</td>
<td>ARG2</td>
<td>ARG3</td>
<td>ARG0</td>
<td>ARG1</td>
<td>ARG2</td>
<td>ARG3</td>
</tr>
<tr>
<td>Total Count</td>
<td>2</td>
<td>883</td>
<td>4244</td>
<td>6219</td>
<td>6671</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>% N</td>
<td>-</td>
<td>-</td>
<td>0.02</td>
<td>17.91</td>
<td>99.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>% of Total N</td>
<td>-</td>
<td>-</td>
<td>0.01</td>
<td>14.23</td>
<td>85.75</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>% V</td>
<td>100.00</td>
<td>99.32</td>
<td>88.10</td>
<td>13.76</td>
<td>0.10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>% of Total V</td>
<td>0.04</td>
<td>16.00</td>
<td>68.22</td>
<td>15.62</td>
<td>0.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>% A</td>
<td>-</td>
<td>-</td>
<td>7.07</td>
<td>64.14</td>
<td>0.03</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>% of Total A</td>
<td>-</td>
<td>-</td>
<td>6.99</td>
<td>92.96</td>
<td>0.05</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ gnab := n\_c\_le \& \{ STEM < "gnab", SYNSEM \{ LKEYS.KEYREL.PRED "arg0\_n\_1\_rel", PHON.ONSET con \} \}. \]

Figure 3: A Sample Lexical Template

\[ arg0\_n\_1\_rel : ARG0 \_x. \]

Figure 4: A Sample Predicate-Argument Template

Now that we have a basic notion of the contents of the lexicon, we can schematize the parsing process as follows:

1. The LKB is given a sentence to parse.
2. ERG matches the input words with their lemmas\(^6\) from the lexicon.
3. The LKB tries to relate these words in a manner that matches the relations licensed by the grammar rules in ERG.
4. All the valid relations are described in the standard MRS format.
5. That format is translated into frames native to SNePS.

3. FEATURES INTRODUCED BY SFY

In pulling together the above systems to create Sfy, we have had to make a handful of alterations to the off-the-shelf distributions. We will also describe some of the code written to interface the many facets of Sfy.

3.1 Changes to LKB/ERG

In order to maintain the default ERG lexicon and predicate-argument definitions separate from those unique to Sfy, we have added two new files for the LKB to load: `sneps-lexicon.tdl` and `sneps.smi`, respectively. Initially, both of these files are empty. We will discuss the how, when, and what of adding to them in the methods section.

We also created a file for lexical item templates (`sneps-template-lexicon.tdl`) and predicate-argument definition templates (`sneps-templates.smi`). These are used in the shotgun approach (see below) to simulate generic lexical items in parsing. For instance, the nonsense word “gnab” in figure (3) has a stem form “gnab”, uses the semantic predicate `arg0\_n\_1\_rel`, and begins with a consonant. From figure (4), we can see that the predicate `arg0\_n\_1\_rel` takes a single argument.

3.2 Changes to SNePS

SNePS is in the process of acquiring native support for WordNet \(^1\). Through these system hooks, SNePS will have

\(^6\)A word’s lemma is the canonical form. This is often identical to the dictionary form.
access to all the semantic and relational information currently encoded in the WordNet database. The current sophistication of our algorithm relies primarily on synonym relationships.

As LKB and ERG use MRS as their native form for semantic encoding, Sfy must also translate these flat predicate structures into a format SNePS can use in reasoning. The representations’ formal equivalency is addressed by Shapiro [7]8. Informally, we have listed the two new frame descriptions and their semantics in figure (5) that help provide native support for MRS. Every time a predicate is used, it must also have a frame associated with it. In practice, this means that any new predicates (i.e., one unseen by SNePS so far) have a frame created for them on the fly as they are produced by the LKB.

4. UNFAMILIAR LEXICAL ITEMS
It is easy to take a prescriptive viewpoint when developing any system dealing with natural language. According to such a mindset, it is very important for the system to distinguish between those sentences licensed by the language and those which are not. With the standard LKB distribution, any sentences not licensed by ERG fail to produce a parse tree. In developing Sfy, we have taken a very descriptive viewpoint on the input language. From a descriptive viewpoint, if LKB fails to produce a parse tree then either our lexicon or grammar rules are insufficient to explain the data. Coping techniques for incomplete grammar rules are beyond the scope of this paper. The look-up and the shotgun approaches exemplify the great flexibility possible with the combination of LKB and SNePS for coping with missing lexical items and senses.

4.1 The Look-up Approach
The classic look-up approach requires comprehensive electronic sources. WordNet will serve as our token external source until sufficient testing can shed light onto the preferred features of these secondary data inputs. Informally, lexical items not found in ERG’s native lexicon are lemmatized9. Given the citation form of a word10, we use system hooks for our external source (e.g., WordNet) to collect the appropriate semantic and/or syntactic information for our parser. Finally, we create a new lexical entry based on that information and re-parse the sentence with this new knowledge integrated into the LKB. Formally, it is much more difficult.

The first problem we find is that WordNet does not provide the fine-grained POS distinctions ERG requires (see table (1) for examples). In fact, it is highly unlikely that any source other than ERG will provide the right POS types.

Next, even with a proper POS type, we need a semantic predicate to associate with the word. Luckily, ERG does not require fully specified semantics at the word level11. This gives us some flexibility as to what predicates we can use. One option is to use the semantic predicate associated with a more general term for the target word as a base value. We would then need methods for further specification. A second option is to pull the predicates of the target word’s synonyms. Presently, there does not seem to be any strong evidence for one approach over the other. We have chosen to use the latter option with the hope that multiple synonyms will provide more fertile data to construct a template from. In fact, we have also coopted the synonyms’ POS type to help determine a fitting POS for our unknown word. Thus, the synonyms provide a POS type, predicate template, and predicate argument structure with which we can construct a new entry for our target word. For a thorough example, see the next major section.

4.2 The Shotgun Approach
In contrast with the rich information needed for the look-up approach, the shotgun approach assumes incomplete information. When a human comes across a new word, she may not be conscious of deciding its part of speech. Nonetheless, this largely syntactic information is quite essential to any further inquiry. In a realistic model, we need a back-off method for words not found in the traditional sources. The shotgun approach assumes we can relax our notion of POS type to allow our lexical item to exist in a sentence as any and all parts of speech. Only some of this overgenerated list of sentences will be parsable. From the valid parses, we can then tentatively adopt the POS type of the mystery lexical item in that particular sentence. In other words, an unknown target presents a syntactic hole to our parser. Only certain parts of speech will fill that hole. We can try to validate all the possible fillers by naively testing every part of speech. The subset of sentences the LKB can parse informs us of exactly which parts of speech our unknown word can be.

In terms of Sfy, the lexical entry and generic predicate structures function as those slot fillers. Assume a world in which a reader is constantly coming across new names of people and new verbs. Naively, he could replace any unknown word with a name and a verb he does know. More often than not, only one of the two sentences will make sense. This allows terms follow the most standard patterns of conjugation or inflection. On the other hand, some irregular or erroneous forms may crop up in the data. We do not yet have a good solution to all such cases.

7For our purposes, a synonym is WordNet’s notion of multiple terms that share a gloss.
8See [6] for a more thorough treatment of the subject.
9A lemmatizer takes as input an arbitrary word and returns its lemma. For instance, Lemmatize(elephants) and Lemmatize(elephant) should be the same. Likewise, Lemmatize(is) and Lemmatize(am) should be equivalent.
10Reasonably, we cannot always expect our word to be readily lemmatized. On the one hand, the most productive new
noname := n_-_pn_le &  
  [ STEM < "noname" >,  
    SYNSEM [ LKEYS.KEYREL.CARG "noname",  
      PHON.ONSET con ] ]  
  
Figure 6: The Proper Noun Template

kim := n_-_pn_le &  
  [ STEM < "kim" >,  
    SYNSEM [ LKEYS.KEYREL.CARG "kim",  
      PHON.ONSET con ] ]  

Figure 7: Lexical Entries for “Kim” and “elephant”

our reader to assume the unknown word is of the same part  
of speech as the successful replacement word. In the world  
of Sfy, we follow the same formula but have to use replace-  
ments that match every POS. In traditional grammar, that  
means we use a noun, verb, adjective, adverb, etc. In the  
LKB, we need to use as many POS types as ERG contains  
(see table (1) above).

5. SFY IN THE WILD

Using a hybrid example from the LKB literature and SNePS  
literature, we will show some of the flexibility of Sfy in  
this section. The simple sentence in example (1) poses no prob-  
lem to the LKB because it is a fully licensed sentence. The  
lexical entries for both nouns (shown in figure (7)) are part  
of ERG’s base lexicon. Likewise, the sentence matches the  
grammatical specifications of ERG. Examples (2) and (3)  
provide two examples of ungrammatical sentences. The  
former example fails because there is not proper agreement  
between the plural “Kims” and the singular “is”. In the latter  
example, the common noun “elephant” cannot stand alone  
without a determiner (e.g., “the” or “an”).

(1) Kim is an elephant.
(2) [*]Kims is an elephant.
(3) [*]Kim is elephant.

More interestingly, we need to try sentences that contain  
words outside of ERG’s lexicon. Example (4) contains the  
unknown word “harbinger”. Manually looking up “harbinger”  
in WordNet, we find it has both noun and verb synonyms.  
In table (4), we have listed the synonyms by part of speech.  
The columns show which words also contain entries in the  
ERG lexicon. Given that all three noun synonyms have the  
same POS type, it is safe to assume that “harbinger” would  
also have this POS type. We can likewise deduce the argu-  
ment structure from the synonyms.

<table>
<thead>
<tr>
<th>Nouns</th>
<th>Nouns</th>
</tr>
</thead>
<tbody>
<tr>
<td>forerunner</td>
<td>herald</td>
</tr>
<tr>
<td>herald</td>
<td>precursor</td>
</tr>
</tbody>
</table>

Unfortunately, only two of the verb synonyms exist in the  
ERG lexicon. They also do not agree with respect to their  
POS type. Moreover, “announce” appears as two separate  
entries with different POS types. For now, Sfy uses all three  
forms as templates for three new verb entries for “harbinger”  
in the SNePS lexicon file. Eventually, we may need to re-  
move erroneous “harbinger” POS types if we find they are  
not confirmed in any input data.

(4) Kim is a harbinger.

Next, there are those sentences which contain words in nei-  
ther ERG nor WordNet. The name “Clyde” and the common  
noun “oliphant” both fit these requirements. To properly  
parse example (5), Sfy must first find the lexical entry tem-  
plates that make a grammatical sentence. Of the two most  
common POS types for nouns shown in table (1), only the  
proper noun variant is viable. We can test this intuition  
by replacing “Clyde” in the input with “noname”, the faux  
lexical item we created with the entry in figure (6). Unfor-  
utunately, Sfy’s shotgun approach is not yet sophisticated  
enough to handle multiple unknown lexical items. Thus, if  
we try to parse example (6) before example (5), the LKB will  
not be able to create a possible parse. On the other hand,  
by parsing the sentences in the order shown, a new lexical  
entry for “Clyde” will be created after parsing example (5)  
which we can use in parsing example (6).

(5) Clyde is an elephant.
(6) Clyde is an oliphant.

5.1 Further Work

We have only begun to take advantage of the many strengths  
of LKB and SNePS alone and as a pair. For instance, SNePS  
is an ideal environment to recreate the semantic hierarchy  
of WordNet. Combining this ontology with ERG’s types,  
type hierarchy, and predicates, we could begin making in-  
ferences about unknown words based on their position in the  
overlap between the two systems. Another important step  
in developing Sfy is supplying the semantic predicates with  
word-level meaning. One approach to this problem is to use  
LKB to parse the glosses provided by WordNet for lexical  
items. Interestingly, the words would be defined exactly by  
their use in the parser.

12The asterisk is used in Linguistics to mark sentences that  
are ill-formed or ungrammatical.

13An archaic form of the more common “elephant”
Next, ERG’s coverage is well documented. Corpus coverage and other long-term studies will provide important information as to what new territory can be analyzed with the help of Sfy. As mentioned above, Sfy currently overgenerates lexical entries. In combination with these same corpus studies of which words and senses are most commonly used, rules for rolling back or removing lexical entries will be very useful in future versions. In fact, because the techniques described above are purely lexical in nature, we could be oversupplying lexical entries for what is really a hole in the grammar rules. Again, coping techniques for an insufficient grammar is beyond the scope of the current work.

As a related sub-problem to the shotgun approach, we may need to add a new part of speech to a word already in our lexicon\(^{14}\). While there are many subregularities in the meaning change associated with a part of speech change, we have no guarantee an algorithm could uniformly handle such productive variation. Thus, we could not rely on a post-hoc reanalysis to recast the target word into a new POS. The shotgun approach could be augmented to provide for these cases when the orthographic form of a word is known but the particular syntactic use is novel.

In conclusion, the lexical acquisition problem is just one of many interesting facets that need to be addressed.

6. REFERENCES

\(^{14}\)It is quite plausible that a known word will be used in an unfamiliar syntactic role.